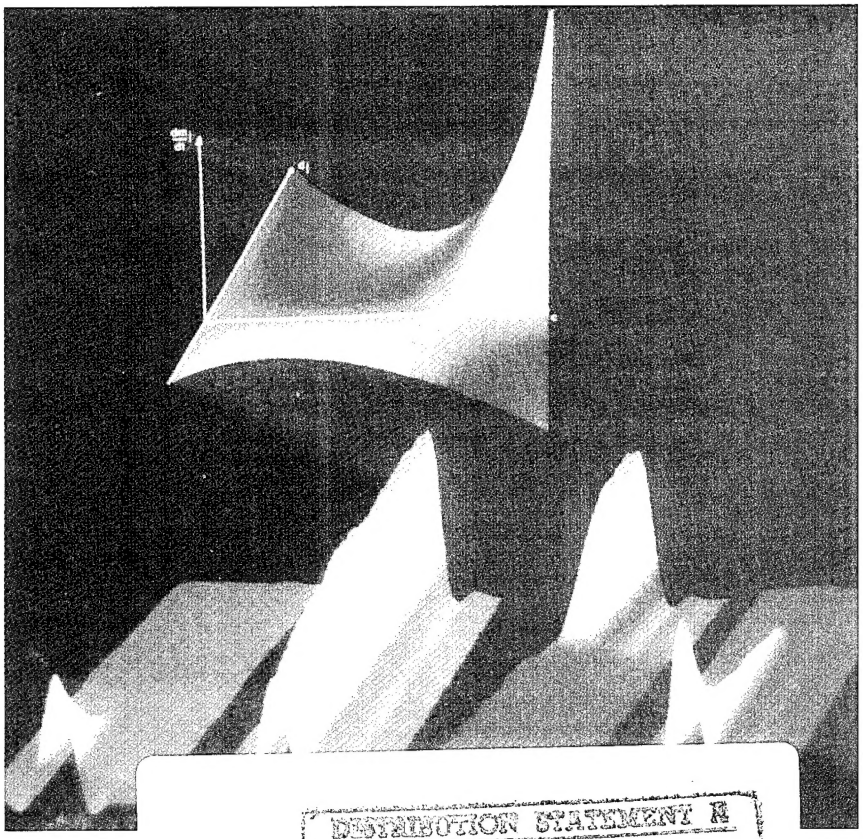
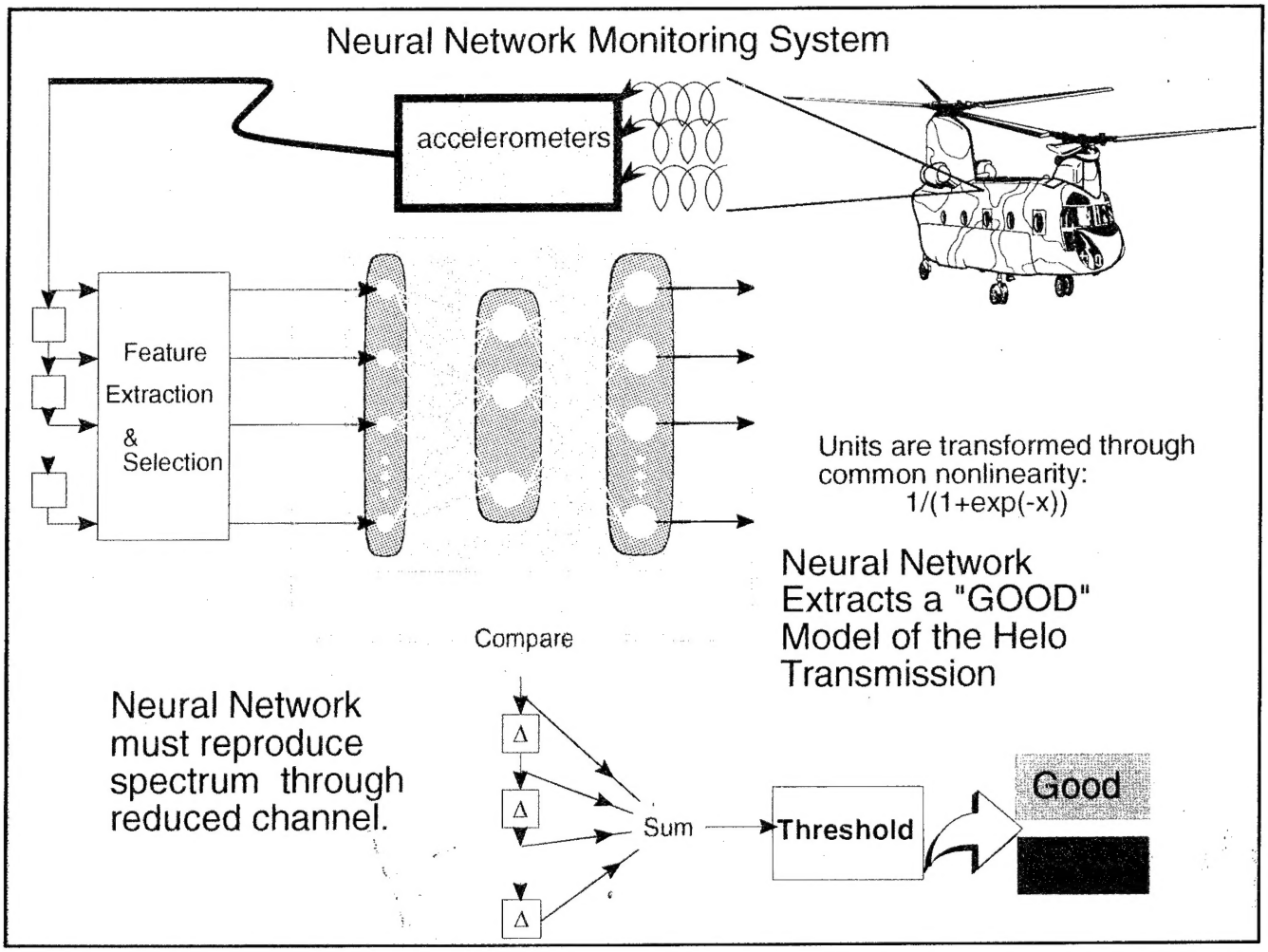
Naval Research Reviews

Office of Naval Research
Three / 1995
Vol XLVII



Science to Applications
A Decade of Neural Network Research



Helicopter transmission monitoring for predictive failure. The first successful use of a neural network to diagnose a previously unknown fault in an operational Navy aircraft was achieved recently. The basic concept behind this Office of Naval Research supported research is to use a model of the hippocampal (pertaining to a specific section of the brain) processes for recognition memory and to train the model to recognize the normal range of vibration or noise signals from a mechanical device. *See article on page 3.*

Naval Research Reviews

Office of Naval Research
Three/1995
Vol XLVII

Articles

Departments

About the Cover

Drs. Bienstock, Cooper, and Munro (Brown University) have suggested a testable theory (BCM rule) to describe how connection strengths between individual calls are strengthened or weakened. This theory can be ussed to describe connection strengths (i.e. learning) in real brain cells or in abstract neural networks. These figures describe and predict the changes in connection strength between cells in a part of the brain known to be required for normal vision. The top figure represents how sign and magnitude of connection strength change for different frequencies and cell responses in the system. This process is inhibited (lower left) and eventually disappears if normal input is removed from one side of the visual system. The second trace is a noraml control. The lower right displays a simulation of visual system data as visual inputs to an eye is eliminated (last trace on right) or newly presented. Applications of the biologically denied BCM rule will be found in the article by Glier, Laird, and Cooper in this issue (see page 5).

Naval Research Reviews publishes articles about research conducted by the laboratories and contractors of the Office of Naval Research and describes important naval experimental activities. Manuscripts submitted for publication, correspondence concerning prospective articles, and changes of address, should be directed to Code OPARI, Office of Naval Research, Arlington, VA 22217-5000. Requests for subscriptions should be directed to the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20403. *Naval Research Reviews* is published from appropriated funds by authority of the Office of Naval Research in accordance with Navy Publications and Printing Regulations. NAVSO P-35.

Introduction

Joel Davis, Guest Editor
Office of Naval Research

This issue of *Naval Research Reviews* represents the third opportunity for me to "show case" the outstanding work being performed by some of our Office of Naval Research investigators in the area of neural networks. In 1985, I predicted in this journal [37 (4): 1985] that biologically motivated neural networks were shortly to be developed ("We are at the point where realistic and powerful models of cellular information processing networks underlying learning and memory can be developed"). Six years later in this journal [43 (4): 1991], I presented work from three laboratories well on their way to precise functional descriptions of computational neural processes.

In this edition, I have chosen ONR funded contributors who are making the transition from research and development (R&D) to product. These are some of the first, successful attempts to apply neural network algorithms to the solution of real world problems. Dan Hammerstrom, one of the contributors to this volume, has eloquently written about the difficulty in hardware implementation of neural networks. Although neural networks are an extremely powerful set of techniques that are being extensively used in the real world and have become invaluable as a paradigm for research into human intelligence and biological computing structures, they have not yet revolutionized human/computer interfaces and intelligent computing. One response is, of course, that this field, if not in its infancy, is probably only in early adolescence. A second response is that neural networks currently being used constitute a part of a larger system. Optical Character Recognition (OCR) is a good example. Most commercial OCR systems use neural network classifiers, but other, more traditional, computational networks are added to the system. These "hybrid" systems may represent the near term applications avenue.

Another applications problem involves the analog vs. digital issue. More has been written on this topic than could be covered in this forum. Whereas, analog networks hold out the promise of ultra-low power operation which will enhance down-sizing and portability constraints; these systems are currently hard to mass produce and to get to work right over a range of temperatures, voltages, and other operating conditions. I would like to suggest that these problems are solvable. In fact, ONR is taking the lead in examining these shortcomings of silicon analog systems.

This issue of *Naval Research Reviews* features three articles from contributors who have bridged the gap from basic research to technology applications. All of them have been participants in the ONR Small Business Initiative Program. Mr. Vincent Schaper and Mr. Doug Harry of the ONR Industrial Outreach Division should be commended for their vision and actions which have stimulated members of the small business community to take an active role in neural network applications. These young entrepreneurs are an integral part of transitioning this technology to the Navy and the marketplace.

Another article describes a very thrilling Navy-relevant neural network application that may have already saved lives in an operational situation. The article describes the work of Professor Mark Gluck, currently at Rutgers University, and Mr. Robert Kolesar at Naval Command, Control and Ocean Surveillance Center in San Diego and the application of a biologically inspired algorithm to diagnose faults in a helicopter gearbox. The network has its roots in the basic research on animal and human learning performed by Dr. Gluck more than a decade ago as a post-doctoral student at Stanford supported by an ONR grant to his mentor. At first glance, the connection between animal learning and defective helicopter gear boxes may be difficult to make. However, a strong basic research program requires a strategy that is willing to see connections and take chances for the first time.

Rear Admiral Marc Pelaez, Chief of Naval Research, has noted that "The Navy in recent times has taken a lot of hits for supporting such basic research as neural research ... if you were to just look at a research abstract, you might not understand why the Navy would be funding such work, but this neural net (Dr. Gluck's novelty detector) is the result of exactly that sort of research which has come under question, and we're still only elucidating a small piece of the potential applications."

The human brain and its functional abilities represent the ultimate in evolutionary processes that began millions of years ago. Understanding how learning and memory takes place is one of the most important basic questions still to be answered in the life sciences. I believe that neural networks will allow us to organize our neuroscience knowledge, suggest new testable hypotheses, and allow the fruits of that research to enhance the Navy's mission. I hope this issue of *Naval Research Reviews* conveys some of the current excitement in the neural network applications area.

Mechanical Fault Detection System

A neural network mechanical fault detection and classification technique developed by Mark A. Gluck, an assistant professor at the Center for Molecular and Behavioral Neuroscience, Rutgers University, Newark, recently proved its effectiveness when it confirmed a faulty transmission in a Marine Corps CH-46 helicopter. This project is funded by the Office of Naval Research (ONR). Gluck is a former recipient of ONR's young Investigator Award, which is given each year to a promising young researcher.

When an unknown anomaly first appeared in data gathered on this presumed problem-free helicopter initial analysis proved inconclusive. Gluck's neural-based technique was put into operation and quickly and accurately indicated that a problem did exist. The helicopter, which had just participated in a major fleet amphibious exercise, was taken out of service and its aft transmission removed. An engineering analysis of the transmission revealed three possibly serious gear faults that had gone undetected by the aircrew.

"These faults were significant enough to pull the helicopter out of operation to prevent any other problems that might have translated from the faults," said Rear Admiral Marc Pelaez, Chief of Naval Research. "Although the neural-net was only in the testing stages, this maturing technology already has proven it's capable of outperforming conventional methods of testing."

According to Pelaez, the Navy expects to have neural-based systems installed on a helicopter by this spring as part of the Air Vehicle Diagnostic System (AVDS) Advanced Technology Demonstration to demonstrate their effectiveness as early warning systems.

"This is a whole new approach to fault detection with tremendous commercial applications," he noted.

The fault detection and classification system analyzes vibrational signals given off by helicopter gearboxes to determine the health of the component. An important feature of this neural-based diagnostic technique is the hippocampal (a specific section of the brain)-based network developed by Gluck. Gluck's neural-based network will, for the first time, allow characterization of mechanical faults that are heretofore unknown. This feature is called "novelty detection."

Gluck's hippocampal-based network has also been used to detect and classify both faults in aircraft carrier fire pumps as well as sonar signals.

Besides having the potential of saving lives through an important adjunct for safety through early and accurate detection and classification of faults, the neural-based system also is expected to significantly reduce operations and maintenance costs for the Navy and other potential users, said Pelaez. Instead of overhauling equipments on a "time-based" schedule, the neural-based system should allow for "conditioned-based" maintenance, whereby machines are repaired or reconditioned only when there is objective evidence of failure, he said.

Development of the neural-based detection and classification technologies incorporating Gluck's hippocampal algorithms has taken place at the Naval Command Control and Ocean Surveillance Center in San Diego, under the direction of Robert Kolesar, Head of the Advanced Development Group.

The hippocampal-based technology developed by Gluck consists of computer programs using the basic hippocampal learning process common to humans and other animals. Key to this system is the essential component of memory formation and recognition based on past experiences and information to determine the relative value of new and different cues and inputs.

For example, before flight the hippocampal network is taught to learn the vibrational patterns from "good" gearboxes to determine a range of normal operations. Once the system has been trained on a wide variety of "good" data, the hippocampal model is brought into play to detect and classify anomalous indications falling outside of the range of "goodness." The system works by assigning values (numbers) to various inputs based upon whether they fall into the range considered normal.

"The system, in essence, comes to learn what a normal gearbox sounds like, so it can then identify an abnormal vibrational pattern when it is inputted," explained Gluck. "It's the same principle as someone knowing what his car engine sounds like, and being able to identify when something is wrong because the engine sounds different. No one else may be able to hear the difference, but the owner of the car has come to 'learn' what sounds normal, and is able to determine when something is wrong."

Unlike the human mind which can follow only about four to five streams of information at a time when analyzing complex problems, the hippocampal-based network is capable of handling streams of information numbering in the hundreds, providing the potential for analyses previously outside the range of human ability and knowledge. Besides its applications in mechanical fault determination, the hippocampal-based network holds the potential of being utilized in such fields as medical diagnoses and economic forecasting.

The hippocampal-based network has its roots in the basic research on animal and human learning performed by Gluck more than a decade ago as a post-doctoral student at Stanford University. In his initial research, which was supported by ONR funding, Gluck worked with rabbits to see if they could be taught to learn how to blink their eyes according to specific

patterns, and how that learning took place. From there, his research moved to working with individuals who suffered from amnesia and hippocampal damage to determine more clearly the hippocampal-based processes involved in memory formation and recognition.

In addition to the development of neural-based systems for analyses and detection, Gluck's research also paves the way for improved understanding of both normal and abnormal functions of the human mind. For example, it may become possible to use his research for the design of learning tasks that would pinpoint the location of damage to the brain that is the source of different types of learning difficulties. With these sorts of indicators in hand, it then may become possible to develop better diagnostic and treatment methods to assist those who suffer from the mental disorder occurring as a result of such problems as Alzheimer's disease, schizophrenia, strokes, and traumatic brain injuries.

Neural Network Applications Using the Ni1000 Recognition Accelerator

Michael Glier and Mark Laird, Nestor, Inc. and Dr. Leon Cooper, Brown University

Editor's Note

The fastest neural network processor, the Ni1000 Recognition Accelerator (computer chip), was developed recently by Intel Corp. of Santa Clara, California, and Nestor Corp. of Providence, Rhode Island, with Office of Naval Research and Advanced Research Projects Agency (ARPA) funding. The new technology is the most promising approach toward building intelligent machines that mimic hearing, seeing and thinking. The chip is amazingly quick at recognizing handwriting, identifying military targets and performing other tasks that are difficult or impossible for conventional chips. There are numerous civilian applications for this chip, including finger print identifications, automatic mailing address processing, and even stock market forecasting and predicting.

These chips work more like the human brain then the microprocessors used in millions of personal computers. Because they can recognize visual or sound patterns at high speed, neural nets are being applied to tricky tasks such as distinguishing human voices and zip codes. ARPA is interested in these chips for identifying submarines and other targets.

Nester developed a version of the handwriting-recognition algorithm for the Ni1000. A scanner based on a fast version of Intel's 486 microchip can recognize about 30 handwritten characters per second while the Ni1000 is expected to recognize 5,000 to 10,000 characters.

Where other chips answer precise mathematical questions, neural net chips can be trained to work on more subjective problems. Interconnected processing elements on each chip, called neurons, join in different ways when exposed to different signals. By employing a large number of processing elements that operate in parallel, the Ni1000 performs 25 billion interconnection operations per second. The chip uses a large block or flash memory so that learned patterns can be "memorized"

Introduction

For many years computation has been dominated by considerations of speed and efficiency since complex, time-consuming calculations had to be accomplished with limited means. We now have to adjust to an era of plenty, in which vastly increased memory and processing power makes use and convenient access more important than speed and raw computational power.

In a curiously parallel fashion, neural networks have also been dominated by computational constraints; in some of these networks, learning is very slow and in all of them the intrinsically parallel computations are not efficiently executed on serial machines. This time of constraint may now be ending.

Recently, Intel and Nestor Corporations, under Advanced Research Projects Agency (ARPA) and Office of Naval Research (ONR) contracts, have produced the Ni1000, a 3.7 million transistor VLSI chip that has 1024 sparsely connected neurons of 256-input dimension with 64 outputs. This chip has an on-board RISC processor and on-chip learning. It can perform up to 16 billion integer operations or about 33,000 classifications per second and is expected to provide real-time performance in various military and commercial applications.

Since many Ni1000 chips can run in parallel and since future generations of hardware will no doubt increase the number of neurons while decreasing power needs and cost, we are entering a time of hardware plenty for neural networks. Rather than struggling to make neural networks small and less complex, and rather than conserving the number of neurons employed, we may in the future be primarily concerned with ease of use, accuracy and ability to generalize.

Biology provides us with an example, the brain. With this instrument, we manage with stunning success (at least occasionally) to recognize patterns and make rapid, if sometimes incorrect, decisions in complex real-world situations.

The Ni1000 Recognition Accelerator

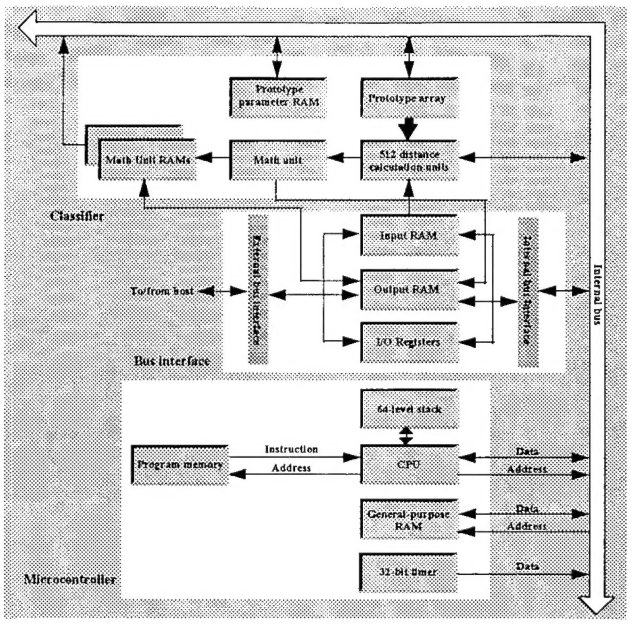
The Ni1000 recognition accelerator is a high performance radial basis function (RBF) neural network chip. The chip offers the ability to accelerate neural network applications with performance equivalent to up to 16.5 billion operations on a general purpose microprocessor.

The Ni1000 design is shown in Figure 1. It consists of a parallel, pipelined radial basis function neural network, made up of 3 independent functional units:

- A radial basis function neural network *classifier*
- 16-bit RISC *microcontroller*
- A *bus interface*

Figure 1.

Block diagram of Ni1000 Recognition Accelerator Chip



The Radial Basis Function Neural Network Classifier

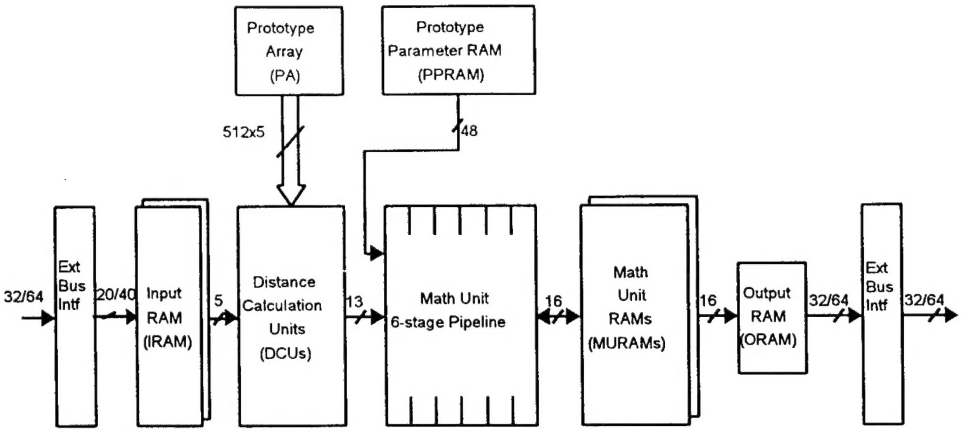
The classifier contains:

- 512 parallel distance calculation units
- A prototype array memory, which can store up to 1024 prototype vectors in on-chip FLASH memory
- A 6-stage pipelined math unit

The 1024 prototype by 256 feature array memory is organized such that two prototype vectors are associated, and physically adjacent to, one of the 512 distance calculation units (DCU). Each of the DCUs first calculates the distance between the input feature vector and one of the DCU's two local prototype vectors. This distance is transferred to the math unit. Then the DCU calculates the distance between the input feature vector and its other local prototype vector, if necessary. This second operation does not occur if fewer than 500 proto-

Figure 2.

Ni1000 Data Flow.



types are in use. Instead, the next vector begins processing, which increases throughput.

The math unit produces deterministic results (a list of firing classes) and 16-bit floating point probabilistic results (6-bit exponent, 10-bit mantissa) in parallel. It has a 6K RAM for storing two full sets of results for all 64 classes, both deterministic and probabilistic. The math unit can accept one new distance on every clock.

A 16-bit RISC Microcontroller

The on-chip, custom, 16-bit RISC microcontroller has separate program and data memories. The 4K x 16-bit nonvolatile FLASH EPROM memory can hold training algorithms, chip maintenance utilities and other software required by the application. A general purpose 256 x 16-bit RAM is also accessible to the microcontroller. A set of standard microcode is provided with the chip that implements a simple race-free interface protocol and two training algorithms, Restricted Coulomb Energy (RCE) and Probabilistic Neural Network (PNN).

It is also possible to do all of the training external to the chip, then load the trained network parameters into the Ni1000. Functions are provided in the standard microcode to facilitate loading data into the chip. Functions are also provided to copy data from the chip, which can be used to replicate the trained network in other chips.

The Bus Interface

The Ni1000 bus interface provides double buffers on the input to permit pipelining of input vectors. An external master can access the chip's I/O registers to control and monitor the classifier, and to communicate with the microcontroller.

Operation

At 33 MHz, the Ni1000 can classify over 32,000 input vectors per second, where each input vector has up to 223 features, each with 5-bit resolution. This performance level is made possible by the Ni1000's parallel architecture, which executes up to 16.5 billion operations per second. A typical Von Neumann machine would need to execute more than 65 billion instructions per second to approach the processing rate achieved by the Ni1000 Recognition Accelerator.

The Ni1000 data flow is shown in Figure 2. The Ni1000 supports all neural network learning on-chip, via an embedded microcontroller. Because the chip includes this embedded microcontroller, the user does not need to be a neural network expert to implement a complete neural network system. Training can be accomplished by simply presenting the patterns and their corresponding class labels to the chip.

Separate dual-input data buffers and a single output buffer are provided. This permits simultaneous pipelined operation on up to three input patterns. The output buffer provides several output data formats to support various application requirements, including integer and single-precision IEEE floating point.

Since prototypes are stored in nonvolatile FLASH memory, no off-chip prototype memory or performance-stealing prototype loading operations are needed. The chip stores approximately 6 Kbytes of prototype parameter data in its on-chip RAM. This is stored in RAM since it must change during the training process. However, once the training process is completed, this data can be made nonvolatile by copying it into reserved FLASH. A microcontroller firmware routine is then used to copy the FLASH data into the prototype parameter RAM each time the chip is powered up.

The low hardware overhead required to incorporate the chip into a system is further enhanced by the fact that no

external boot or program memory is required, since the internal microcontroller's program memory is also FLASH memory.

The in-circuit reprogramming of the Ni1000's FLASH memory also provides a mechanism for enhancing the chip's learning algorithms in the field, reducing the cost of upgrades and maintenance, and preserving the customer's investment.

Application #1: Optical character recognition

Several manufacturers of document image processing equipment are developing new, low cost document readers using the Ni1000. Nestor itself is developing a high-speed image processing and recognition system for high-volume "smart-forms" transaction processing. Called NiReader, it will feature Ni1000 chips as its core recognition processing elements.

Such products are expected to dramatically improve both the speed and the accuracy of the processing of hand-printed forms, significantly reducing the cost of paper work for business and government. For example, in health-care systems smart forms are beginning to eliminate administrative bottlenecks, dramatically reducing the time required for the collection of patient information and for the payment to providers.

Application #2: Mail Sorting

A manufacturer of mail sorting equipment has developed a low-cost envelope reader which reads the destination zip code from the printed address field on an envelope, then prints a postal bar code on the envelope. Businesses can save up to 10 cents per item when mailing bulk mail by using bar coding techniques. The Ni1000 permits this reader to process mail at up to 50,000 pieces per hour, dramatically reducing bulk mailing costs.

Initial testing on one bulk mailing achieved 99.7% accuracy on Canadian (alphanumeric) zip codes. Although the board's interface could not support the required data rate, the theoretical classification rate for the Ni1000 on this problem was 143,000 characters per second.

Application #3: Intellient Forms Processing (N'Form)

Nestor's N'Form is a document identification and routing product based on the Ni1000. N'Form features robust and accurate identification of document form types, including multi-part forms, without any required pre-processing to correct for skew or document rotation. The prototype version already exhibits high accuracy forms recognition of dozens of form types under difficult conditions, including 180 rotation, notes in margins, significantly defaced forms and even stick-on notes.

N'Form will deliver important benefits to high volume document scanning operations, including an estimated 4x increase in scanner operator productivity, support for fast scanning job changes, fully integrated forms processing, including Intelligent Character Recognition (band printed) (ICR), Optical Charater Recognition (machine printed) (OCR), and Optical Mark Recognition (OMR) and bar code recognition plus automated document indexing and routing.

N'Form will automatically perform key image quality assessment functions, including orientation detection (for correction of documents imaged upside down or rotated), blank form detection (to identify forms that have not been filled out) and blank page detection (to correctly detect the true form presented as a sequence of two images from a scanner operating in duplex mode). Single or multi-part documents can be routed either to a user- specified directory associated with the form type, or to another application for next stage processing.

A separate Document Administrator module will support forms training capability directly from images of sample forms. In addition to initial training, this permits training on new forms and on images that are rejected by the system.

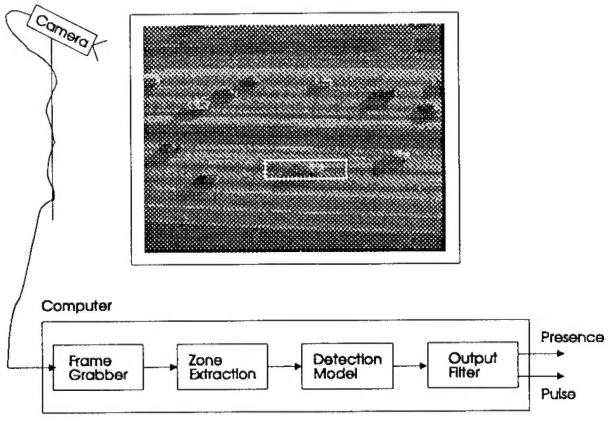
The product, targeted for availability by mid-1996, will support real-time forms identification operating at 120 images/minute from uncompressed or TIFF G3, 2d or G4 compressed images.

Application #4: Traffic Monitoring

Vehicle detection systems using video technology are currently being deployed. Neural networks can improve on

Figure 3.

Operation of a typical video detection system.



these systems through more robust detection (e.g. bicycles, pedestrians and emergency vehicles) and an ability to classify vehicles (or other objects). Figure 3 shows the operation of a traditional video-based vehicle detection system. Figure 3. Operation of a typical video detection system.

A system using vehicle classification can *track* the vehicle (or object) from camera to camera. Traffic engineers and traffic control systems can use this information for intelligent *corridor management*, and law enforcement can track vehicle movement to avoid high speed chases. Figure 4. Detection model architecture.

Figure 4 illustrates the role of the neural network in the detection system. The trained network indicates the position of the vehicle in the image by producing a Gaussian shaped curve (Figure 4) where the peak of the curve indicates the longitudinal center of the vehicle within the detection zone.

The Ni1000 can support video data classification from up to 4 standard CCTV cameras at 30 frames/second. This information processing rate permits the Ni1000 to capture and classify multiple frames containing the same vehicle to obtain a high confidence classification.

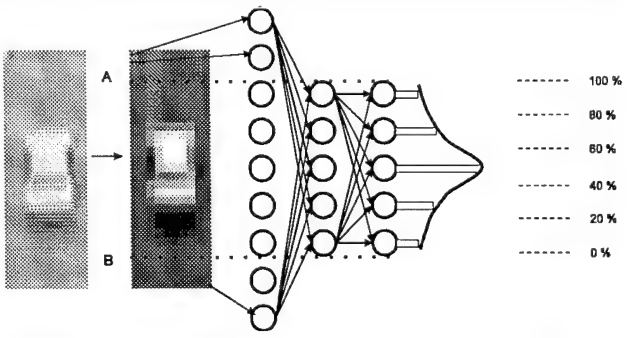
In recent tests at a live traffic signal installation at Louisiana State University's Remote Sensing Laboratory, the Ni1000 achieved 92% accuracy when used as a vehicle presence and passage detector in day, night and dawn/dusk transition conditions. For comparison, an existing highly-optimized and dedicated image processing system has up to 95% accuracy under full daylight or night time conditions, but exhibits significantly degraded accuracy during dusk and dawn transition periods, when traffic is often at its peak.

Application #5: Vision systems

Neural-network-based visual inspection systems can quickly "learn" to recognize flaws in new products (or new versions of existing products), without the need for specialized programming or time-consuming system adjustments. Quality assurance can be fine- tuned, and instantly adjusted as products and markets evolve. Neural-network-based intelligent inspection systems are in use today in pharmaceutical, automobile and even potato chip manufacturing.

Figure 4.

Detection model architecture.



Application #6: Fingerprint Matching

Current fingerprint identification methods use high-cost mainframe-based systems that are too large and expensive to deploy in local police departments. As a result, automated fingerprint identification can involve long time delays - as long as one week for certain FBI files. In addition, their accuracy, when matching partial fingerprints, is unsatisfactory

One manufacturer has developed an Ni1000-based fingerprint matching system that promises to move this capability to local police departments. This will allow rapid identification of suspects, reducing the time that known criminals remain unidentified.

These Ni1000 based systems will allow local law enforcement officials rapid analysis, based on readily available digitized fingerprint databases, and will also provide improved accuracy from matches of partial prints - all at a fraction of the cost of existing systems. A similar device for digitized photo matching is also being developed.

Ni1000 Specifications

The Ni1000 comes in a 168-pin CPGA and can operate from 10 MHz to 33 MHz in the commercial temperature range. The part can also operate over an extended temperature range, with some reduction in maximum operating frequency. When running at 33 MHz, it dissipates 3 watts at 5V, and requires 30 mA of +12V during programming or erasure of the FLASH memories. Its I/O signal levels are TTL compatible.

At 33 MHz, it classifies a minimum of 32,000 patterns per second when employing all available input features and prototypes. Smaller classification problems can be run substantially faster. In fact, some real applications that have been run on the Ni1000 execute at well over 100,000 patterns per second.

The chip itself contains over 3.7 million transistors on a 13mm x 15mm die. It is fabricated by Intel using their 0.8 m CHMOS-IV process.

Ni1000-Based Products

A development system is available for developing solutions using the Ni1000. ISA, PCI and VMEbus compatible boards are currently available and Ni1000s can be purchased separately for custom board designs.

Summary

Neural networks have evolved into high performance, low-cost and easy-to-incorporate solutions. Using the Ni1000, they can deliver capabilities formerly limited to supercomputers, on a PC or workstation.

Applications developers are rapidly learning about the capabilities of neural networks, and are beginning to employ them in hundreds of applications. The dramatic success stories of Ni1000 customers prove the wide applicability of this technology.

Integration times (often only a few days to a few weeks, when replacing other software or hardware neural networks) demonstrate the ease of integration of the hardware and software.

Acknowledgment

The authors wish to acknowledge funding by ARPA and ONR that made the development and deployment of the Ni1000 possible. In addition, we acknowledge the significant contributions and work of Darcy Bullock at LSU in application of the Ni1000 to traffic monitoring applications.

Biography

Dr. Leon Cooper is co-chairman and founder of Nestor, Inc. and Thomas J. Watson Senior Professor of Science at Brown University as well as the Director of the Institute for Brain and Neural Systems there; he has more than 25 years of neural network experience. Professor Cooper's work in theoretical physics ranges from the foundations of the quantum theory to low-temperature physics; he has also done theoretical work in modeling neural networks as well as work on the molecular basis of learning and a memory storage. Professor Cooper was awarded the Comstock Prize by the National Academy of Science in 1968 and the Nobel Prize in Physics in 1972. He is a Fellow of the American Physical Society and American Academy of Sciences, and member of the National Academy of Sciences and the American Philosophical Society, and is the author of more than 100 published scientific works and eight US patents.

Michael Glier is Vice President of the Embedded Systems Division at Nestor and is responsible for the development of products that utilize the Ni1000 and commodity hardware to accelerate the Company's document processing products and pattern recognition software. Mr. Glier provided the architectural direction and managed the successful implementation of the ARPA funded Ni1000 project with Intel Corporation. Mr. Glier was co-recipient of the 1994 Discover Award in Computer Hardware and Electronics for the effort. Mr. Glier has over 28 years experience in management and development with the electronics industry, from space-based systems to multiprocessor design and deployment. Mr. Glier is the author of two patents and several technical articles.

Mark Laird is a Project Engineer at Nestor, Inc. and the Principal Investigator for Nestor's development of software and add-in boards for PCs funded by ARPA and the Office of Naval Research. Mark has 19 years of experience in developing computer systems' hardware and software from mainframes to PCs, the last three at Nestor developing neural network solutions. His article published in VEMBus Systems magazine in June, 1995, provides additional information on the Ni1000 and on a VMEbus compatible board based on the Ni1000.

Neural Network Applications Speed The Navy's Warfighting Ability

A.J. Maren, R.M. Pap, K.L. Priddy, and R. M. Akita
Accurate Automation Corporation

Editor's Note

This article describes the successful results of two Navy programs working together. First, funding came from the Office of Naval Research's Cognitive and Neural Science and Technology Division; these funds were then channeled through the Navy's Small Business Innovation Research (SBIR) program to Accurate Automation Corporation (ACC), a small business enterprise in Chattanooga, Tennessee, which has successfully developed novel neural network technologies for the Navy. With the help of the SBIR program, ACC has grown from a basement shop of two people in 1987 to 20 people today in its own building of 10,000 square feet; revenues have grown during the same time from $18,000 to $4,000,000.

ACC has developed among other technologies for the Navy "Neural Network Toolbox" and the Sparse MIMD Neural Network Processor, which will make a PC computer work as fast as a super computer.

Mentioned in this article are the three phases of funding managed by SBIR to promote small business. In phase 1, up to $100,000 is awarded to a small business for 6 months to evaluate the technical merit and feasibility of an idea; phase II awards up to $750,000 for two years to expand the results of phase 1 by developing a product; and phase III allows for the commercialization of the product through a "buyer" in the government or private industry.

Thus by choosing small companies with the right capabilities for the job, the Navy can improve the scientific as well as the business strength of the nation.

Introduction

There's an old adage among'st fighter pilots, Speed is life, more is better. These pilots spoke only in reference to their platforms. In today's world, the need for speed applies to all areas of warfighting.

The German army, during the 1940's, mastered the art of "Blitzkrieg," or "lightning warfare," which referred to the technique of rapidly building for and executing an attack. Today, all areas of warfare demand speed and precision. This applies not only to the abilities of the forces in direct contact, but also - and most especially - to functions that support the warrior in combat, including sensor fusion, data communications, database management, target and image recognition, and rapid surveillance. These capabilities directly influence our ability to carry the battle to the enemy, and are the critical elements in our ability to strike an early, crippling blow.

According to Adm. William A. Owens, Vice Chairman of the Joint Chiefs of Staff, "Read the flagship pronouncements of each of the military services: The Army's description of *Force XXI*, the Navy's *Forward ... from the Sea*, the Air Force's *Global Reach, Global Power*, and the Marine Corps' *Operational Maneuver* ... from the Sea. The visions they sketch are remarkably similar. Each points toward the capability to use military force with greater precision, less risk, and more effectiveness. Each relies on three areas of technology:

- Intelligence, surveillance, and reconnaissance,
- Advanced command, control, communications, computers, and intelligence, and
- Precision-guided munitions.

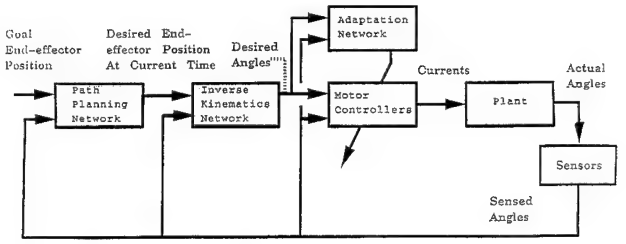
Each recognizes that its efforts are a part of a broader undertaking. I believe that this is the U.S. revolution in joint military affairs."

Neural networks are a key enabling technology that will enhance all three of these critical technologies. Today's battlefield, whether on air, land, or sea, uses a number of neural network hardware and/or software applications. For example, all modems make use of neural network technology for adaptive echo cancellation [1]. Many of the neural network innovations are inspired by biological neural networks. The diversified interplay between different neural network research programs has led to a strong basis for technology development and transition to fielded use by the Navy.

To meet these Navy needs, we have been developing novel neural network technologies that will support and speed the Navy's warfighting capabilities on many levels. These technology developments, sponsored by the Navy Small Business Innovation Research (SBIR) program, include neural adaptive control (leading to advanced flight control methods), sensor fusion and figure-of-merit determination as well as data compression and automatic target recognition. Neural networks apply to Militarily Critical Technologies [2] such as sensor fusion and signal processing, hypersonic / waverider

Figure 1.

Overall controller system design.



aircraft design and control, simulation / visualization methods, and intelligent processing equipment. In the latter area, in order to develop a platform that can give the Navy maximal computational speed, Accurate Automation Corporation (AAC) has developed a Multiple Instruction, Multiple Data (MIMD) *Neural Network Processor* (NNP^{TM}).

The following is a summary of AAC's recent developments in neural network technology for diversified Navy applications.

Neural Adaptive Control

An autonomous control system is one in which the system itself generates the appropriate control action and/or trajectory. An autonomous controller, whether used in robotics or for flight control, should use desired position values in relation to current position to determine the appropriate motion within a range that is dictated by the capabilities of the plant. Adaptive control systems are those systems that change their own parametric structure to compensate for changes in the plant being controlled.

As part of ongoing work in adaptive control, AAC has developed novel neural network methods for inverse kinematics determination, under a Phase II SBIR contract funded by the Office of Naval Research (ONR). Our *Neural Network Processor* has been used to solve the inverse kinematics problem in near-real-time. These solutions were tested on a real robot using a VME card cage and a dual DSP (TI-TMS 320-C40) implementation attached to the *Neural Network Processor*. The tests were run at NASA Marshall Space Flight Center, AL, using the Proto Flight Manipulator Arm or PFM. In addition to the inverse kinematics using neural networks, a unique joint controller [3] was developed using the functional link neural network paradigm. The overall concept of this is shown in Figure 1.

The inverse kinematics problem can be explained as a set of coupled equations which couple joint parameters to the desired end effector trajectory. Thus, by solving this set of

coupled equations we can determine the desired joint moves to obtain the desired trajectory. The solution of optimization problems has been found using recurrent networks [4-8] for a variety of applications. We will show how the solution of sets of equations using a linear Hopfield network can also be modified to solve the inverse kinematics problem.

Classical manipulator kinematics describes the position of an end effector based upon the positions of the joints of a robot arm. Generally, this is given by a set of non-linear equations, f(.) in joint space, (θ).

$$y(t) = f(\theta(t)) \tag{1}$$

What we really want to know in the inverse kinematics case are the desired joint positions, (θ), to obtain y(t). The inversion of f(.) is difficult due to the dimensionality, multiple joints, and the inherent non-linearities found in robot motors.

$$\theta(t) = f^{-1}(y(t)) \tag{2}$$

A common method to facilitate the solution of the inverse kinematics problem is to linearize the forward kinematics, i.e. to differentiate f(.) with respect to time, yielding the velocity equation (or differential kinematics)

$$\frac{dy}{dt} = \frac{d}{dt}[f(\theta)] = J(\theta)\frac{d\theta}{dt} \tag{3}$$

where J (θ) is the Jacobian of θ.

The prescribed trajectory y(t) is then tracked by a linear approximation via inversion of the linear velocity equation and integrating the obtained joint angle velocities.

Using this approach, the inverse kinematics problem reduces to the inversion of the Jacobian matrix **J**[θ(t)] at all time instants along y(t), yielding

$$\frac{d\theta}{dt} = J^{-1}(\theta)\frac{dy}{dt} \tag{4}$$

The solution for the inversion of the Jacobian is obtained using a combination of a feedforward neural network and a linear Hopfield network. The Hopfield neural network solves an equation of the form

$$x(i+1) = Wx(i) + u \tag{5}$$

where

W is a symmetric (n x n) real or complex weight matrix,

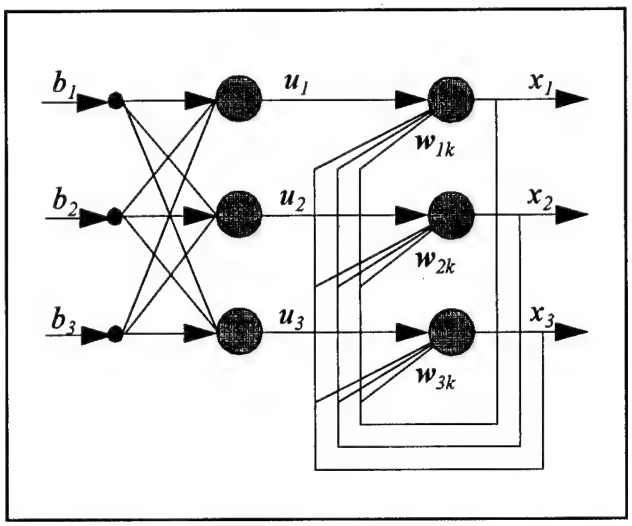
u is a real or complex n-vector of inputs, and

x(i) is the ith iteration of the n-dimensional vector of neuron states.

The resulting equation converges to a solution when the spectral radius, ρ(W), is less than unity, i.e. the eigenvalues of

Figure 2.

A hybrid linear dynamic network, comprised of a feedforward layer followed by a linear Hopfield network.



W lie within the unit circle in the complex plane. When solving equations of the form Ax = b, the required finesse is to set the weight matrix and input for the Hopfield network to

$$W = I - \alpha A^H A \tag{6}$$

$$u = \alpha A^H b \tag{7}$$

where

A^H is the Hermitian (complex conjugate transpose) of A.

These equations converge for

$$0\,\alpha\,\frac{2}{\rho(A^H A)} \tag{8}$$

In order to speed computations the spectral radius is often replaced by the trace which is the upper bound for the spectral radius.

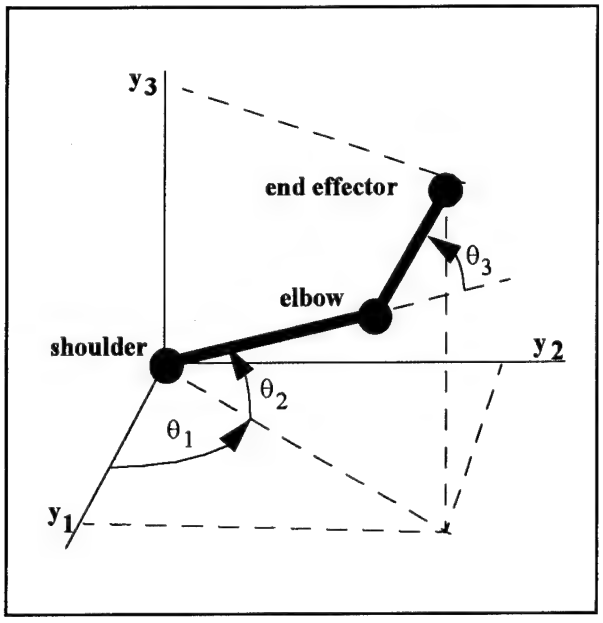
$$0 < \alpha < \frac{2}{trace(A^H A)} \tag{9}$$

The neural network implementation of the linear Hopfield solution for a system of equations of the form Ax = b is shown in Figure 2.

The inverse kinematic problem is solved by forming a difference equation for (t) at a given discrete time along the

Figure 3.

The robot manipulator model used for analysis of the inverse kinematics problem.



y(t) trajectory. The required solution is then used to move the joints to the desired position until the next time increment. Typically, the solution is obtained in a few hundred iterations of the Hopfield network on an NNP which is much faster than real-time to a robot joint.

$$\Delta\theta\,(i+1) = W\Delta\theta\,(i) + u \tag{10}$$

where

u is a real or complex n-vector of inputs held constant at time t.

$\Delta\,\theta$ (i) is the ith iteration of the n-dimensional vector of neuron states for time t.

The first step is to determine the values for the weight matrix and the input, u, for the Hopfield neural network. From the previous explanations it is fairly straightforward to see that the weight matrix and input to the Hopfield network for the linearized method of computing the desired joint positions are given by

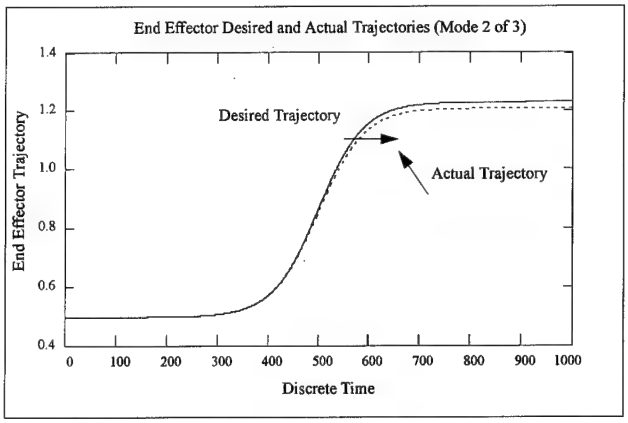
$$W = I - \alpha J^H J \tag{11}$$

$$u = \alpha J^H b \tag{12}$$

$$0 < \alpha \frac{2}{trace(J^H J)} \tag{13}$$

Figure 4.

Desired and actual end-effector trajectory in Cartesian space (1000 points). The error of mode 2 vanished when the trajectory consisted of 1500 points.

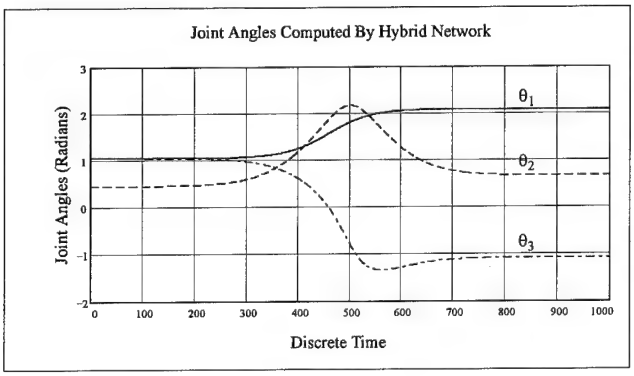


which yields the "best" solution for $\Delta\theta$ for each time step in a least squares sense. Once we find the solution for $\Delta\theta$, we issue incremental changes to the joints which maintains the desired trajectory.

The hybrid feedforward-Hopfield neural network is capable of solving the inverse kinematics problem in real-time when implemented on the AAC *Neural Network Processor* (NNP™). The Hopfield neural network was shown to converge to an optimal solution in a least squares sense and is applicable to a variety of optimization problems.

Figure 5.

Joint angle trajectories computed by the hybrid feedforward/linear Hopfield network.



This capability simulated the motion of a three joint robot arm on a Silicon Graphics 4D380VGX superminicomputer as depicted in Figure 3. The arm was given a straight line trajectory in Cartesian space consisting of 1000 points and 1500 points. The trajectory for the 1000 point case was identical to the desired with one slight exception in one of the dimensions as depicted in Figure 4. The 1500 point case was exact in all three dimensions. The joint angles generated by the hybrid neural network are shown in Figure 5.

As a result of the success in the inverse kinematics project, the concepts were applied to flight controls for LoFLYTE.

Neurocontrol for the Low Observable Flight Test Experiment (LoFLYTE) Aircraft

The Low-Observable Flight Test Experiment (LoFLYTE) Advanced Technology Testbed Aircraft is being built to demonstrate neural network technologies in a real world aircraft using some of the Navy-funded SBIR research at AAC. LoFLYTE is laying the foundation for developing next generation aircraft. The LoFLYTE hypersonic aircraft, shown in Figure 6, is a research test vehicle to investigate performance of a waverider aircraft shape at hypersonic velocities (regime of Mach 5). A waverider is an aircraft which rests on its own shock wave as it flies. Waveriders such as LoFLYTE would typically utilize SCRAMJET engines.

Hypersonic aircraft, such as LoFLYTE, could be used in a number of applications. First, they could be configured as an unmanned surveillance platform. Due to their high speed, they will be able to make passes over areas with minimal concerns about being shot down. This will greatly increase our surveillance capabilities during hostilities. Second, a hypersonic cruise missile could be configured with a capability to reach enemy targets much faster than conventional cruise missiles. The need for rapid attacks is supported by A. Fields Richardson (Capt. USN, Ret.), who, during Desert Storm, served as Principal Navy Liaison to the Joint Forces Air Component Commander and head of the Navy Strike Cell. According to Mr. Richardson, "The ability to strike quickly and with great precision is critical to tactical and strategic success. To launch a cruise missile or an aircraft at a target 1,000 miles away and receive a Battle Damage Indication (BDI) report in 20 minutes will provide dramatic tactical advantage, enabling an irresistible build-up of momentum for the force possessing that capability."

Finally, hypersonic vehicles could be configured as a manned aircraft to be launched from the decks of aircraft carriers.

The LoFLYTE project, funded from an Air Force Phase II SBIR Contract, is the focal point and primary demonstration vehicle for several Phase I and Phase II hypersonic-related SBIR contracts funded by the Air Force and by NASA.

The primary focus of AAC's ONR-sponsored research, as it relates to this project, has been to apply lessons learned from biological neural systems to accurate real-time motor control. This research has led to a number of technologies applicable to the real-time control of complex systems, ranging from robotic manipulators to helicopters to hypersonic aircraft.

Real-time control of such complex systems has required not only new, rapidly adaptive neural network algorithms, but also hardware which can carry out the necessary computations with great speed. The ONR-sponsored work has led directly to development of the AAC *Neural Network Processor* (NNP^{TM}) discussed in the next section. This processor makes possible the extensive and rapid computations necessary to control a hypersonic aircraft that will fly at Mach 5. In addition to the NNP^{TM}, AAC has developed, partially under ONR sponsorship, a full Toolbox of neural networks and learning methods. Several Toolbox neural networks, including the Adaptive Critic [9], have been successfully applied to the control of complex systems at AAC. The Toolbox is central to the development of the control algorithms being used in LoFLYTE. The motor control and motor-mapping algorithms developed for ONR are other examples of Navy technology which have formed a basis for portions of the LoFLYTE aircraft control design.[

AAC has developed neural network control algorithms inspired by drive-reinforcement theories of animal learning and adaptivity. We have applied these methods to the control of electrical motors in robotics tasks. By building upon these algorithms, we have created an adaptive actuator controller for LoFLYTE which adapts to changing loads on LoFLYTE's tiperons and rudders. We have built upon insights into how the

Figure 6.

The AAC LoFLYTE waverider aircraft.

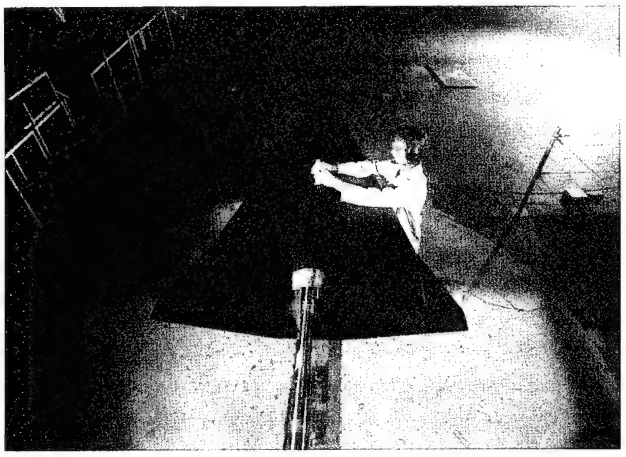


Figure 7.

The AAC Neural Network Processor (NNP™).

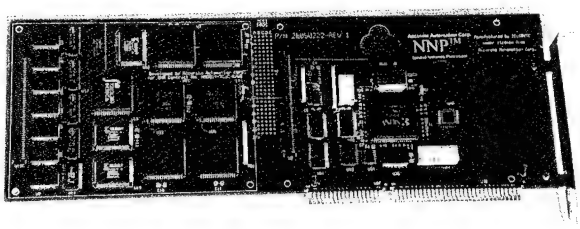
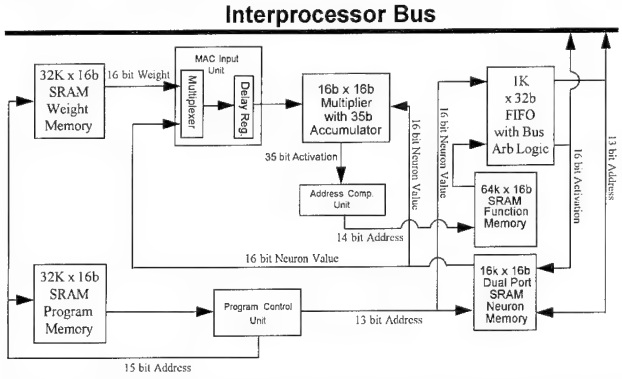


Figure 8.

Block diagram and interprocessor bus architecture for the AAC NNP™.



human brain maps control objectives to desired joint motions to develop new neural network-based methods for mapping flight control objectives to actuator commands.

Multiple Input, Multiple Data (MIMD) Neural Network Processor

As an ongoing part of the research conducted under sponsorship by the Naval Research Laboratory and by Naval Command, Control, and Ocean Surveillance Center, RDT&E Division, Accurate Automation has developed a digital Neural Network Processor (NNP™) which is capable of true parallel processing.

The underlying philosophy in the design of the sparse Multiple Instruction Multiple Data (MIMD) NNP™ has been to achieve maximum computational efficiency in both a single processor and multiprocessor environment by optimizing the design to compute neuron values very efficiently [10, 11]. This is in stark contrast to previously proposed neural network processors which are typically based on classical Single Instruction Multiple Data (SIMD) matrix/vector multiplication architectures. Our design fully exploits the intrinsic sparseness of neural network topologies. Moreover, by using a MIMD parallel processing architecture, one can update multiple neurons in parallel with efficiency approaching 100% as the size of the neural network increases.

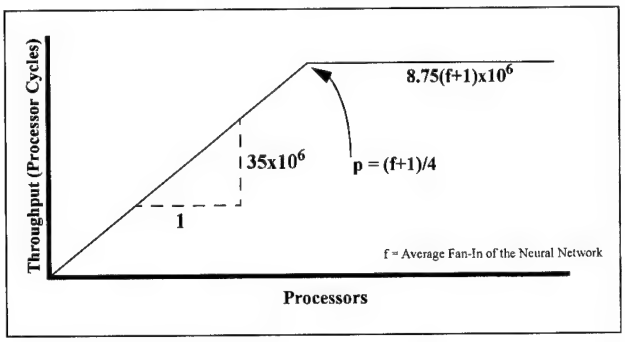
To achieve the desired efficiency we have adopted a design which: 1) Uses an instruction set optimized for neural network processing, allowing one to compute a neuron activation without arranging the weight matrix into linear arrays and/or inserting artificial zero-weighted connection 2) Uses a MIMD parallel processing architecture to permit neurons with totally different input topologies to be updated simultaneously without loss of efficiency; and 3) Uses dual neuron memories to virtually eliminate memory contention and maintain absolute memory coherence.

The NNP™ (see Figure 7) is capable of implementing 8K total neurons with 32K interconnections per processor. A fully configured PC version of the NNP™ is capable of interconnecting 8 modules for a total of 8K neurons and 256K interconnections. In addition to the PC version, the VME version is mated with two TI TMS320C40 DSPs which allow for real-time data manipulation and processing. The VME card is capable of stacking three modules with additional modules requiring a separate VME card due to power limitations. The NNP™ design is based upon a linked list concept which allows any neuron in the network to be connected with any other neuron. Thus, any of the recurrent networks, such as the Hopfield network, or any feedforward networks such as the multilayer perceptron, can be easily implemented using this processor. This hardware capability is vitally important in control applications because it gives us a neural computation engine which is easily adapted to changing control inputs and boundary conditions.

A block diagram of the NNP™ architecture is given in Figure 8. The program instructions and associated weights, as needed, are stored in memory. For a given operation, the instruction is decoded and the necessary value fed from neuron memory to the multiplier input unit along with the weight value the two values are fed to the Multiply ACcumulator (MAC), and the result is passed as an address to the function memory. The address is then used to fetch the appropriate value from the transfer function. This result is then passed though a First-In-First-Out (FIFO) unit and stored in the buffer memory for each of the modules via the interprocessor bus. When an "interchange neuron and buffer memory" (inbm)

Figure 9.

NNP™ throughput as a function of additional processors.



instruction is encountered, each processor finishes processing its code segment before the buffer and neuron memories are interchanged. Once the memories are interchanged, processing continues until another inbm or stop instruction is encountered. The processor also has the ability to multiply two neuron values together. This is done by passing the previous neuron value encountered to the MAC as an input and then passing the current neuron value as an input to the MAC followed by a multiplication operation. The result is passed through the transfer function and stored in buffer memory as explained previously.

The fully pipelined implementation of the neural network architecture delivers nearly one instruction per cycle which allows the neural network board to execute nearly 35 million instructions per second. Using the standard definition of a connection, a byte-wide multiply-accumulation, the NNP™ yields over 140 million connections/sec for a single board. As additional modules are added, the speed increases linearly with over one billion connections per second possible with a full complement of eight processors.

One of the most vital aspects of parallel computing is the ability of the architecture to maintain memory coherence. The AAC NNP™ accomplishes memory coherence through the use of two separate memories, combined with a special interchange instruction. In most neural network implementations, the results from one layer are multiplied by a series of weights summed, and then passed through a transfer function, typically a sigmoid function, before being used as an output for the neurons on the next layer.

In the NNP™, the inputs are read from neuron memory, with the outputs from the transfer functions stored in buffer memory. When all of the neurons on a layer have been processed an interchange buffer and neuron memory instruction is issued which makes the new data available for use by the next layer of neurons.

A particular NNP™, in a worst case scenario, takes four clock cycles to access the bus and write a neuron value to the buffer memory. On the average, each processor would need to access the bus every (f+1) clock cycles, where f is the average fan-in to a given neuron in the network. Thus, the number of processors allowed before contention occurs is $p < (f+1)/4$. When $p > (f+1)/4$, then bus contention occurs. A depiction of the NNP™ throughput as additional processors is added, shown in Figure 9. As can be seen in the figure, the throughput increases linearly as processors are added until $p > (f+1)/4$.

The inherent speed of the NNP™ in processing sparse matrices makes it ideal for computing neural network structures such as the cooperative-competitive neural network, described in the next section.

Sensor Fusion

One of the primary objectives of our work has been to develop new sensor data fusion capabilities for the Navy. Sensor fusion is a Militarily Critical Technology [12-14]. Neural networks present a method by which sensor fusion systems can learn from experience, instead of always requiring explicitly the *a priori* probabilities that are currently needed for existing (e.g. Bayesian) formalisms. Further, neural networks have the potential to give a system adaptability to changing environments and conditions. Finally, neural networks can be implemented in exceptionally fast parallel-processing hardware (such as the AAC NNP™), thus overcoming the huge computational burden associated with real-time sensor fusion. Such a neural network-based capability can play a crucial role in enabling the Navy to build integrated systems, linking together systems which are currently "stovepiped."[

One of the big challenges in sensor data fusion is assigning each target to the right track. Gates can be used to find out what targets are in the vicinity of the expected target positions from different tracks. There is still a problem of conflict resolution, when different targets could be assigned to two or more tracks. We have addressed this challenge by developing the novel COoPerative-COMpetitive (COPCOM) neural network. The neural network determines which items in a given Set A have closest similarity to items in another Set B. This network operates by making iterative target-to-track assignments, so that:

- Targets are assigned where there is maximal closeness between a target and the track across a set of matching metrics, and simultaneously
- Targets are assigned where there is minimal conflict between a prospective match and other possibly competing matches.

In this, it offers a more robust approach to assignment than simple non-optimal assignment methods, and unlike the existing optimal assignment algorithms, it can be implemented in

parallel-processing hardware (e.g. the AAC NNP^{TM}) for real time solutions to the target-to-track assignment problem [15].

The advantage of using the COPCOM network for assignment tasks is that it makes its first (highest strength) assignments to those matches which have the overall highest values in favor of making the assignment (cooperation across multiple dimensions of similarity), and which also have the least competition with other possible assignments. By making the least ambiguous assignments first, the complexity of the overall problem is reduced. This can make it easier to determine future assignments.

The multilayer COPCOM neural network was initially developed to deal with one of the major challenges of image understanding - that of recognizing objects when they are composed of many different parts, are partially obscured, and/or have specular reflections. To meet this challenge, an early, prototype version of the COPCOM neural network identifies those portions in a segmented image which are most related to each other. This early version of the COPCOM neural network has been applied to images where high specularities, dark contrasting shadows, and including objects present substantial challenges for image understanding [16,17]. The COPCOM neural network has since been redefined to create associations between objects in two different sets. This can be used for matching objects, or tracking the evolution of a multipart system over time.

Inspiration for the COPCOM design came from the perceptual psychology of vision, which suggested that many factors, e.g. similarity of intensity, boundary line continuation, and proximity, all played a role in perceptual organization [18,19].

The COPCOM neural network plays a vital role in target-to-track assignment in the AAC Sensor Fusion Tracking System. Preliminary coarse and fine gating produce a string of potential new-target matches for every Master Target Track (MTT). The COPCOM method is then used to resolve matching assignments. A matrix of possible matches to possible tracks is used to partition out the problem, so that only the subset of new detections which can potentially match a given subset of targets is considered at a time. Unique target-to-track assignments are made before the COPCOM network is used. A search of the COPCOM output nodes for those whose activations pass threshold yields an ordered list of non-competing assignments. The pairwise combinations with the strongest activations are listed first. The Tracker uses this output to make assignments, prune the remaining possible target-to-track assignment possibilities, and rerun COPCOM as often as necessary to get a complete set of assignments.

The cooperative-competitive method has been demonstrated to be effective for target-to-track association, even in dense target environments. Some of the scenarios for which COPCOM effectiveness has been shown include crossing targets, splitting targets, and dense targets (e.g. close groups of 4, within overall interacting scenarios of 16 proximal targets) [20-22].

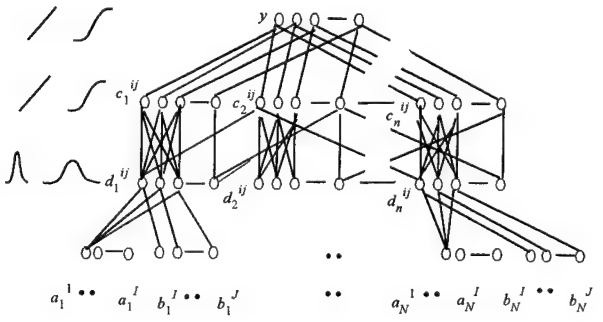
The items to be matched, i.e., the members of Set A and Set **B** (e.g., new detections and tracks), must have the same dimension or vector length, denoted N, and the elements in these two vectors should have the same meaning. (Of course, if matching is being done between elements of the same set, then this requirement is automatically satisfied; Set **A** = Set **B**.) For example, the items of Set **A** and Set **B** could each be the x and y positions of objects in Euclidean 2-space. We denote the nth dimension, of the ith item of Set **A** as a_n^i, and the nth dimension, of the jth item of Set **B** as b_n^j. Let there be a total of I items, i = 1..I in Set **A**, and J items, j = 1..J, in Set **B**. The COPCOM network operates on functions of the distance between the vector components for each possible pairwise match of members of Sets **A** and **B**, over each of the N dimensions describing each set member.

The COPCOM network conceptually consists of four or more layers. A basic COPCOM network is shown in Figure 10. The nodes in Layer 1 represent the individual items themselves. The nodes in Layers 2 and above represent the strength of relationships between pairs of items taken from Set **A** and Set **B**, not to the individual items themselves. This means that if Set **A** has I items, and Set **B** has J, there are I*J nodes in each of N subnets at the second and succeeding layers, to accommodate that number of pairwise relationships.

COPCOM works by assessing relative similarities between items across multiple dimensions. In Layers 2, 3, and subsequent intermediate layers, there is a separate subnet for each dimension which will contribute to the overall assignment decision. Thus, if the items in Sets **A** and **B** can each be described by a 2-D vector (e.g., x and y values), then Layers 2, 3,... will each have two subnets; one for each dimension. We could call these the X subnet and the Y subnet. This paradigm

Figure 10.

The CoOPerative-COMpetitive (COPCOM) neural network architecture.



has been instantiated with up to 5 dimensions. The larger the number of dimensions, the more effective the assignment process is, because more different types of information are used in making assignments.

In the final layer, there is only one subnet. This subnet combines the results of the multi-subnet processes in the lower layers. The details of the COPCOM neural network are fully specified in [20] and results of this network applied to image processing are described in [16,17].

The inputs used in the first subnet on the first layer are all the vector elements. The inputs to the nth subnet on the first layer are element values (e.g. position) in the nth dimension.

The second layer, the **D** or difference layer, of the COPCOM network computes a function of the pairwise differences for each of the N dimensions of the items of Sets **A** and **B** being compared. There are N distinct subnets in the second layer, one for each dimension. For each subnet in the second layer, the value of a node d_n^{ij} is computed as

$$d_n^{ij} = f(a^i_n - b^i_n) \tag{14}$$

where f may be a Gaussian function or an exponential decay function applied to the absolute value of the differences (as is done in the current implementation of COPCOM), or any other monotonically decreasing even function. The result of using this function is that the actual difference between two vector elements is scaled to (0,1]. Thus, when the distance between two items in any dimension is 0, the strength of the node in the subnet for that dimension is 1. As the distance between two items increases, the strength value put into the Layer 2 subnet node decreases towards 0. This pairwise difference metric is computed separately for each dimension.

There are the same number of subnets and nodes in the third and succeeding layers as there are in the second. The only exception is the final layer, which is configured as a single subnet, with the same number of nodes as the subnets in the previous layers. Each node in the third and succeeding layers, the **C** or cooperative-competitive layers, receives both direct inputs from the corresponding node in the previous layer and either cooperative or competitive inputs from other nodes in the previous layer.

For the current implementation of the COPCOM network, the user selects how many cycles of cooperation and/or competition are to be used, and in what order they are to be employed. This is conceptually equivalent to selecting the number of cooperative-competitive layers which will be used in the network. The term cycle refers to a single application of either a cooperative or a competitive process. The connections between cooperative (excitatory) and competitive (inhibitory) inputs are different. If a cooperative cycle is chosen, the nodes c_n^{ij} in the corresponding cooperative-competitive layer receive their activation as

$$c_n^{ij} = d_n^{ij} + E_q \left[\sum_{\substack{i'=1\ldots1 \\ i'\neq i}} d_n^{i'j} + \sum_{\substack{j'=1\ldots J \\ j'\neq j}} d_n^{ij'} \right] \tag{15}$$

If a competitive cycle is chosen, the nodes c_n^{ij} receive their activation as

$$c_n^{ij} = d_n^{ij} - I_q \left[\sum_{\substack{n'=1\ldots n \\ n'\neq n}} d_{n'}^{ij} \right] \tag{16}$$

The connection strength parameters E_q and I_q are chosen by the user, where q corresponds to a given cycle. The version of the COPCOM network implemented in the AAC Tracker allows the user to select as many cooperative and competitive cycles as desired, in any order, and with different weights assigned to each cycle.

The illustration in the accompanying Figure 10 illustrates some of the connections between a d_n^{ij} node and a corresponding c_n^{ij} node. Also, in this figure, both cooperative and competitive links are shown. In actual operation, only one of the two, cooperative or competitive connections, would be used for a d--c transition. The user could also specify additional cycles, resulting in additional layers, each dedicated to either the cooperative or competitive process, and typically alternating.

The dynamics consist of the feedforward flow of activation through the connections described in the preceding paragraphs. At the next-to-topmost layer, the nodes send their activations to the single subnet in the topmost layer, which sums each of the inputs and performs thresholding. The node activations here represent the winners in the assignment process. Connection strengths are typically set before the network is used, and are not adapted during network use.

Biologically-Based Sensor Fusion Circuit

The objective of this work is to create and instantiate a new, biologically-inspired approach to sensor fusion that will have exceptional performance capabilities for the Navy. Using novel, special-purpose circuits, we are creating a parallel processing sensor fusion capability that will emulate the unique multisensor fusion abilities that have recently been discovered to exist in the brain. These include the ability to:

- Correlate detections over a range of spatial registry, thus making possible good sensor fusion even with inexact sensor registry,

- Correlate detections observed at slightly different times, thus enabling combination of sensor data from sensors with different processing speeds or having different times of target observation, and
- Combine weak detections of targets made by different sensors, thus yielding higher probability of detection and lower false alarm rates.

In the course of this work, we are producing a unique sensor fusion circuit based on biological sensor fusion concepts. This new robust and generic sensor fusion capability will have widespread application throughout the Navy, the rest of DoD, and to the private sector. There is currently no similar sensor fusion chip available.

To accomplish this goal, we have formed, under this ONR-sponsored Phase II Small Business Technology Transfer Research (STTR) project, the first teaming relationship between biologists doing sensor fusion research and neural network specialists in sensor fusion who design both hardware and software for the Navy.

We began this work with the premise that biological systems have already solved the sensor fusion problem [23-28]. What is specifically useful is that the biological approach to sensor fusion has already resolved the key issues which press technology development of sensor fusion for DoD. For example, biological sensor fusion is designed to use sensors which are in "loose" registration with each other. This greatly mitigates the need for costly and difficult-to-maintain "tight" sensor coregistration or conversion of inputs into a precisely correlated common reference frame. Biological sensor fusion correlates target observations which appear at slight offsets in both time and space - even though they are of the same target. The biological approach fuses information from different sensors to confirm detection of weak targets, even under conditions of noise. Finally, the biological sensor fusion method has the built-in capacity to do context-dependent orienting and alerting; it can direct attention even to weak but significant stimuli, and it can downgrade response to strong but insignificant stimuli. All of these capabilities are highly desirable, if not necessary, in sensor fusion systems for DoD.

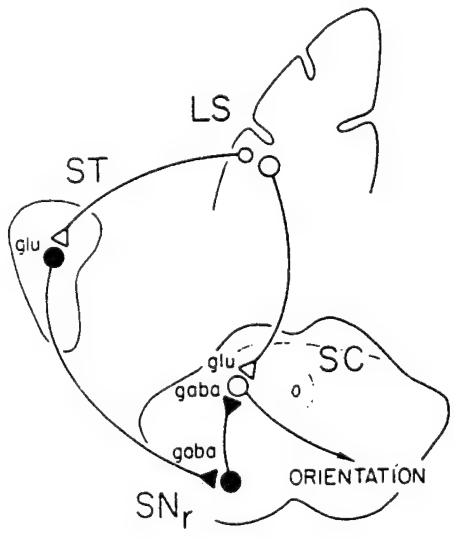
Biological sensor fusion takes advantage of parallel processing and extensive local-neighborhood connectivity. Although software emulations of very simple, scaled-down versions of the biological system can conceivably be hosted in existing serial-processing workstations, the necessary system size precludes advanced sensor fusion system designs. Our calculations indicate that essentially all the memory and computational processes in even a high-end workstation will be used by a full-scale software emulation. This would not allow for design improvements. To overcome this limitation, we have designed specialized biologically-inspired sensor fusion hardware leading to greater down-the-road flexibility and the potential for higher-level performance.

One of the primary characteristics of biological sensor fusion is that it creates (at the cost of great investment of true "neural" wetware) a topographic representation of the sensor-observed surrounding space. This topographic representation is essential to alerting and orienting the animal, and to fusing inputs that are loosely in register with each other. Recent research now indicates that extensive connections from the higher portions of the brain - the striate cortex through the basil ganglia - add the influence of high-level knowledge and context to processing targets in the superior colliculus [29]. Our design preserves these important relationships. Under this project, we are creating a special sensor fusion circuit array that mimics the sensor fusion processing of the superior colliculus. We envision that high-level knowledge stored in programs hosted on a workstation can activate portions of this array, thus assisting target detections.

One of the greatest benefits of this new sensor fusion technology for the Navy is its ability to perform context-dependent target detection. In software emulations of the proposed sensor fusion circuit, we are demonstrating two significant types of embedded context-dependent responses:

Figure 11.

Summary diagram illustrating the presumptive relationships between the "direct" and "indirect" corticotectal pathways. Visual areas of the lateral suprasylvian (LS) cortex provide direct excitatory connections to the superior colliculus (SC). An indirect stimulus comes about from excitatory connections from the LS to the striatum (ST). The excited ST neurons inhibit the substantia nigra pars reticulata (SNR) neurons, which in turn releases the inhibition they have been exerting on (disinhibits) the SC neurons. The indirect route serves as a mechanism for "gain controling" sensitivity of portions of the SC [Figure taken from McHaffie et al., 1993] [30]



- If a target is detected in a given area, look in the neighborhood of the target for other (possibly weaker) targets.
- If a target has been observed in the past, but is now lost, look in the broad neighborhood around its last sighting for a (possibly weaker) detection.

Both of these context-dependent behaviors are embedded into the "wetware" of the biological sensor fusion system, and also are designed into the software emulations and the hardware that we are building. This capability comes about through the "local gain control" which is exerted by the substantia nigra pars reticulata (SNR) on the superior colliculus (SC), as shown in Figure 11.[30]

The context-dependent enhanced target detection capability will be very useful to the Navy. One example of how this capability can be employed is to consider the current status of target tracking in the Detection and Tracking Module (D&T) of the Combat Direction Center (CDC). When a tracked target makes a sharp maneuver, the current trackers tend to lag the radar and Identification Friend or Foe (IFF) returns. The detector/tracker operator could manually intervene to change the nature of the tracker to accommodate maneuvers, especially if the target is in a turn. Occasionally, the radar return may be weak due to the changing aspect of the target. Our novel sensor fusion system has the unique ability to preferentially extract weak detections in the neighborhood of a "lost" initial track. This means that tracks, even of maneuvering targets, will be maintained more readily. Further, if several targets are in the neighborhood of each other, they will be detected more readily if even only one of them gives a multisensor response (or strong single-sensor response). Using the software emulation, we will be able to identify parameters that will give the most desirable response for different operating criteria, thus enhancing the Navy's ability to wage war under diversified field conditions.

Figure-of-Merit Determinations

A figure-of-merit is a metric which estimates the effectiveness or quality of a process or system. One such figure-of-merit which is of great interest in this study is a figure-of-merit in target identification, or FOM_ID. Such a figure-of-merit can help operators know the degree of confidence in a target ID. This knowledge can increase effectiveness by reducing misidentifications and the associated friendly fire incidents.

The FOM_ID is being constructed from two different types of figures-of-merit; one for target identification information, and one in the confidence of uniquely assigning a given target report (including the associated ID information) to a given track. The latter is a unique consideration in developing ID confidence metrics. It will help greatly in alerting operators to possible "ID-rub-off" situations, which have been observed extensively in major exercises (e.g. the Joint Air Defense Operations Joint Engagement Zone (JADO-JEZ) Near-Land 93 Exercise). ID rub-offs can be a major source of ID conflicts, leading to both friendly fire and penetration of Blue regions by hostile forces.

Accurate Automation is conducting research under a recently-awarded Phase II SBIR, under which four useful figures-of-merit will be constructed. These are figures-of-merit in:

- Target Position Error (FOM_TPE) for an individual target, which estimates the error in the position estimate, whether it is constructed using single or multisensor data,
- Assignment Confidence (FOM_ASSIGN), which gives confidence in how uniquely a new target observation can be assigned to a given target track (vis-a-vis other neighboring target tracks),
- Combined INFOrmation on target IDentity (FOM_ID_INFO), which represents the accumulation (over time) of different ID reports for a given target, and their confirmation / disconfirmation of a consistent ID, and
- Target ID (FOM_ID), which represents the combined effect of both target ID information and report assignment (uniqueness of ID assignment) for a given target.

The purpose of a "target position error" figure-of-merit (FOM_TPE) is to assess the error associated with the state position estimate. The FOM_TPE will be useful in determining the accuracy to which a target's position error is known. This will be useful in many ways, e.g., assigning size and duration of scan for scanning radars tracking maneuvering targets, and determining whether the inbound approach of an aircraft is within sensor tolerance for certain types of automated landing, or whether the landing needs to be downgraded. It can be used to assess quality of tracking for weapons control. Further, it sets a technical basis for the next step, developing a figure-of-merit in target-to-track assignment.

The purpose of the new target-to-track assignment figure-of-merit (FOM_ASSIGN) is to answer the question: To what extent can we be assured that the sensor measurements used to update a target track apply to that particular track and not to another? This question must be answered to have effective target ID.

Miscorrelations, caused by sensor information attached to one target track "rubbing off" on a target that comes into neighborhood of the first, are a major cause of target mis-identifications. This can lead to friendly fire, to allowing an enemy to enter secured airspace, and other undesirable events.

The purpose of FOM_ID_INFO is to express the confidence of giving a target classification, both in terms of target type (e.g. fighter / bomber / etc.) and target nature (friend / foe / neutral / unknown). The target classification will be derived from a combination of the different information which confirms / disconfirms target class / ID. It will also be time-vary-

ing, as different information becomes available, or as information which is expected does or does not appear.

Our final step is to construct an overall figure-of-merit for target ID (FOM_ID). This FOM makes use of both FOM_ID_INFO and FOM_ASSIGN. This is the figure-of-merit that will be most directly and frequently useful to operators and tactical commanders in a large number of scenarios, ranging from pilots / RIOs who need to make fire / no-fire decisions, to battlegroup commanders observing targets at a more theatre-level of engagement.

The concept of a Figure-of-Merit is almost universally applicable to assessing system performance. Figures-of-merit have been developed and used for GPS receivers and for sensor performance evaluation. Simple FOMs are currently in use for track quality and for target ID. The neural network figure-of-merit technology is directly applicable to the task of rate illumination [31]. This approach provides a non-parametric means of time-domain and/or frequency domain correlation as it is applied in Naval signal processing systems.

A figure-of-merit formalism for target-to-track assignment was proposed as early as 1980 by [32]. He proposed a simple algorithmic formalism based on distance between the new target observation and the expectation, as well as covariances of past observations. Other formalisms are similarly algorithmic and reliant on simple functions of covariance matrices. AAC's approach to figure-of-merit determination is to investigate the ability of a neural network to learn correlations between the inputs available to a system during operation (sensor observations and predictions), and information available only during training (the difference between state estimate and ground truth). We are conducting experiments to determine the extent to which these correlations can be embedded into the weight structure of a neural network, giving a non-analytic model for performance expectations based on recent observations.

One of the greatest potential uses of figure-of-merit technology is as inputs to intelligent agents. Intelligent agents are a powerful emerging technology which will be used to support individual and collective projects. One of the primary uses of intelligent agent technology for the Navy will be as advisors and/or assistants to battlefield commanders and other military personnel with time-critical missions. It is to be noted that intelligent agents constructed with neural networks could be tailored for each "warrior" to fully support their warfighting needs. Because of the unique ability of the neural net to learn, they can be adaptively trained for each individual warrior's tasks and needs. The goal of these intelligent agents will be to help form and update situation assessments; to rapidly identify those targets and events which will have the greatest impact on their user's mission and safety. Intelligent agents will use figures-of-merit to assess the criticality of different targets, and to assess the impact of target interactions with the user, each other, and their environment. By offloading some of the situation assessment responsibility, and giving the user that information which is most necessary for survival and mission completion, the agents will enable their operators to function effectively in situations which evolve very rapidly and which are confounded by large amounts of available data.

We envision that the intelligent agents that will be built for the Year 2000 and beyond will be systems incorporating aspects of both symbolic computation (e.g., classic artificial intelligence (AI) methods such as expert systems, blackboard systems, embedded domain knowledge, etc.) as well as the more recently evolved "soft intelligent computing" strategies (including neural networks, fuzzy logic, and genetic algorithms). This combination of methods will be necessary so that the intelligent agent can learn in real-time, adapt its rule base to accommodate newly learned rules, build a more complete user and domain model, and adapt to changes in operating conditions.

Directions For Future Work

Application of neural networks to modern warfare requires that researchers work closely with the warfighter and move promising concepts rapidly into fielded use [2]. In addition, researchers must keep in mind that the goal is to insert technology into in-service systems. To this end, Accurate Automation's research efforts are directed towards technology transition via participation in Advanced Concept Initiatives (ACIs). These give the opportunity for rapid concept development, demonstration and evaluation, and integration into Navy warfighting systems. To assist in this mission, AAC develops quality basic research in neural networks that have significant applications potential to a wide range of both DoD and civilian usage. For example, AAC's new "sensor fusion chip," being developed under a new Phase II Small Business Technology Transfer Research (STTR) program, will enable precise multisensor localization of commercial ground vehicles as well as supporting DoD multisensor fusion needs.

Under an NSWC Phase II SBIR Cruise Missile Visualization program, we are developing means to display correlated information in the most useful manner possible. This will allow strike commanders and battlefield commanders to readily visualize the trajectories of proposed retargeting options.

We are investigating methods for object-based intelligent systems development, incorporating both "soft intelligent computing" methods (e.g. neural networks and fuzzy logic) and symbolic knowledge representation. These methods can be used to automatically supplement standard DoD digital databases with the most current information avail-

able. Intelligent agent technology, operating on these geospatial vector databases, will not only identify discrepancies between new information and old, but identify solutions to either update the plan or to request additional information.

Our neural networks are being applied to UHF satellite communication (SATCOM) modems, in a manner that will allow greater data throughput and thus enhance communications for the Navy's FLTSAT in the 25 kHz channel. The modems' performances in terms of bit-error-rate are greatly improved, allowing the higher throughput to occur. This initial modem development replaced the demodulator's integrate-and-dump filter with a neural network matched filter that is functionally equivalent to a combined equalizer, matched filter, and sequential decoder. The neural network based matched filter is matched to spectrally efficient waveforms that have passed through the nonlinear channel. This neural network enhanced modem provides receiver bit-error-rate performance and data throughput not otherwise available on SATCOM links from any fielded communications equipment.

The figure-of-merit development which AAC is conducting under the ONR-sponsored Phase II SBIR contract, described earlier, provides a basis by which intelligent agents can evaluate situations. Embedded knowledge in intelligent agents will allow them to both "fill in" certain aspects of the strike plan (given certain information as starting points) and to identify what information needs to be obtained to complete the revised plan. Intelligent agents can handle some of the query from the strike platforms back to the base, sending and receiving sparsely coded messages that rapidly "fill in the blanks" for the new plan. By shifting some plan completion tasks to agents, the strike aircrews will be able to concentrate their attention on response to the immediate environment.

The combination of these and other technological advances will change the nature of real-time retargeting from concept to reality. This will make possible the most effective use of Navy air assets.

Acknowledgements

The authors describe efforts undertaken at AAC over many years by multiple teams of highly gifted and hard-working people. We particularly wish to note that contributions have been made by Dr. Richard Saeks in the areas of neural adaptive control, particularly for hypersonic aircraft control applications, and in designing and developing AAC's Neural Network Processor™. Mr. Karl Mathia has been instrumental in developing the linear Hopfield method for inverse kinematics used for many control applications. Mr. Chadwick Cox has been lead investigator on numerous neurocontrol projects, and has led AAC's development of an adaptive critic. Ms. Susan Treadway and Ms. Liz McCormick were very helpful in preparing this manuscript.

All research described in this document has been supported by the Small Business Innovative Research (SBIR) program and the Small Business Technology Transfer Research (STTR) program.

Biographies

Dr. Alianna J. Maren is Senior Scientist at Accurate Automation. She received her B.S. in mathematics from the University of North Dakota in 1976, and her Ph.D. in physical chemistry at Arizona State University in 1981. Over the past twelve years, she has specialized in sensor fusion, emphasizing neural network methods for the past eight years, and is currently Principal Investigator on a Phase II SBIR and Phase II STTR, both sponsored by ONR. She is a member of the Board of Governors of the International Neural Network Society, and is lead author of the Handbook of Neural Computing Applications.

Mr. Robert M. Pap, President and Chairman of the Board of Accurate Automation Corporation, received his B.S. in applied mathematics from New York University. He has been Principal Investigator on numerous SBIR contracts as well as a multi-year grant from the NSF. Together with Dr. Maren, he is a co-author of the Handbook of Neural Computing Applications.

Dr. Kevin L. Priddy, Engineering Manager at AAC, received his B.S. in electrical engineering from Brigham Young University in 1982, and his Masters and Ph.D. degrees, both in electrical engineering, from the Air Force Institute of Technology in 1985 and 1992, respectively. He is an expert in neural network applications to pattern recognition, fault diagnosis, and automatic target recognition, and is Principal Investigator on several Phase I and Phase II SBIR contracts. He is, along with Dr. Richard E. Saeks, co-designer of the AAC Neural Network Processor™.

Mr. Richard M. Akita, Vice President - Advanced Technology Programs at AAC, received his B.S. in mathematics from Oregon State University in 1961, and his M.S. in electrical engineering from the Naval Postgraduate School in 1968. Prior to joining AAC, he served as Business Manager, Air Traffic Control and Identification Systems, for Naval Research and Development @Level (NR&D).

References

1. Widrow, B., Glover, J.R., McCool, J., Kaunitz, J., Williams, C.S., Hearn, R.H., Zeidler, J.R., Dongle, E., & Goodlin, R.C. "Adaptive Noise Canceling: Principles and Applications", *Proc. IEEE*, 63 (Issue 12, December), 1292-1716 (1975).

2. The Department of Defense. *Report of the National Critical Technologies Panel*, March (1991).

3. Seraji, H. "Decentralized Adaptive Control of Manipulators: Theory, Simulation, and Experimentation", *IEEE Trans. on Robotics and Automation*, Vol. 5, pp. 183-201 (1989).

4. Hopfield, J.J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proc. of the Nat. Acad. of Sci. USA*, Vol. 79, pp. 2554-2558 (1982).

5. Kelly, M.F. et al. Myoelectric Signal Analysis Using Neural Networks, *IEEE Engineering in Medicine and Biology Magazine*, pp. 61-64 (1990).

6. Mathia, K., Lendaris, G.G., Saeks, R. Linear Hopfield Networks and Constrained Optimization (1994). (Submitted for publication).

7. Mathia, K., Saeks, R. Inverse Kinematics via Linear Dynamic Networks, P*roc. of the World Congress on Neural Networks* 1994, San Diego (1994).

8. Mathia, K., Sacks, R., and Lendaris, G.E., "Linear Hopfield Networks, Inverse Kinematics and Constrained Optimization," presented at the IEEE International Conference on Systems, Man and Cybernetics, San Antonio (1994).

9. Werbos, P.J. Neurocontrol and Related Techniques, in Maren, A.J., Harston, C., & Pap, R.M. (Eds.), *Handbook of Neural Computing Applications*. New York: Academic (1990).

10. Saeks, R., Priddy, K. and Pap, R . "Design of a MIMD Neural Network Processor," *Proceedings of SPIE Conference on Applications of Artificial Neural Networks* V, Vol. 2243, pp. 318-323, April 5-8 (1994).

11. Saeks, R., Priddy, K., Schneider, K. and Stowell, S. "On the Design of an MIMD Neural Network Processor," *World Congress on Neural Networks*, Vol. 2, pp. II-590-II-595, June 5-9 (1994).

12. The Department of Defense. *Critical Technologies Plan for the Committees on Armed Services* United States Congress, May 1 (1991).

13. The Department of Defense. *The Militarily Critical Technologies List*, October (1992).

14. The Department of Defense. *National Critical Technologies Report*, March (1995).

15. Blackman, S.S. *Multiple-Target Tracking with Radar Application*. Norwood, MA: Artech (1986).

16. Maren, A. 'Multilayer Cooperative/Competitive Networks' in A.J. Maren et al. (Eds.) *Handbook of Neural Computing Applications*. New York: Academic (1990).

17. Maren, A., and Minsky, V. Representing the Perceptual Organization of Segmented Images Using Hierarchical Scene Structures, *J. Neural Network Computing*, 1 (Winter), 14-33 (1990).

18. Grossberg, S., and E. Mingolla. Neurodynamics of Form Perception: Boundary Completion, Illusory Figures, and Neon Color Spreading, *Psychological Review*, Vol., 92, pp. 173-211 (1985a).

19. Grossberg, S., and E. Mingolla. Neural Dynamics of Perceptual Grouping: Textures, Boundaries, and Emergent Segmentation, *Perception and Psychophysics*, Vol. 38, pp. 141-171 (1985b).

20. Accurate Automation Corporation. Neural Network Hardware and Toolbox Documentation, Version 0.6. (1995).

21. Maren, A.J., Harston, C.T. and R.M. Pap. A Hierarchical Data Structure Representation for Fusing Multisensor Information, *Proceedings of the Second National Conference on Sensors and Sensor Fusion*, Orlando, (This Proceeding is classified SECRET, the paper is unclassified) (1989).

22. Maren, A.J., Lothers, M. and R.M. Pap. Neural Network Sensor Data Fusion Methods for Naval Air Traffic Control, *Proceedings of the 5th National Symposium on Sensor Fusion*, Orlando, (This Proceeding is classified SECRET, the paper is unclassified) (1992).

23. Meredith, M.A. and Stein, B.E. "Visual, Auditory and Somatosensory Convergence on Cells in Superior Colliculus Results in Multisensory Integration," J. *Neurophysicol*, 56: 640-662 (1986).

24. Meredith, M.A. and Stein, B.E. "Spatial Factors Determine the Activity of Multisensory Neurons in Cat Superior Colliculus", *Brain Res.*, 365: 350-354 (1986).

25. Meredith, M.A., Nemitz, J.W. and Stein, B.E. "Determinants of Multisensory integration in Superior Colliculus Neurons," Temporal Factors, *J. Neurosci.*, 10:3727-3742 (1987).

26. Stein, B.E., Meredith, M.A. "Multisensory Integration: Neural and Behavioral Solutions for Dealing with Stimuli from Different Sensory Modalities." In A. Diamond (Ed.), The Development and Neural Bases of Higher Cognitive Functions, *Annals of the New York Acad. of Sciences*, V. 608, 51-65 (1990).

27. Stein, B.E., Meredith, M.A., and Wallace, M.T. "The Visually Responsive Neuron and Beyond: Multisensory Integration in Cat and Monkey," *Progress in Brain Research*, Vol. 95, 79-89 (1993).

28. Stein, B.E., & Meredith, M.A. *The Merging of the Senses*. Cambridge, MA: MIT (1993).

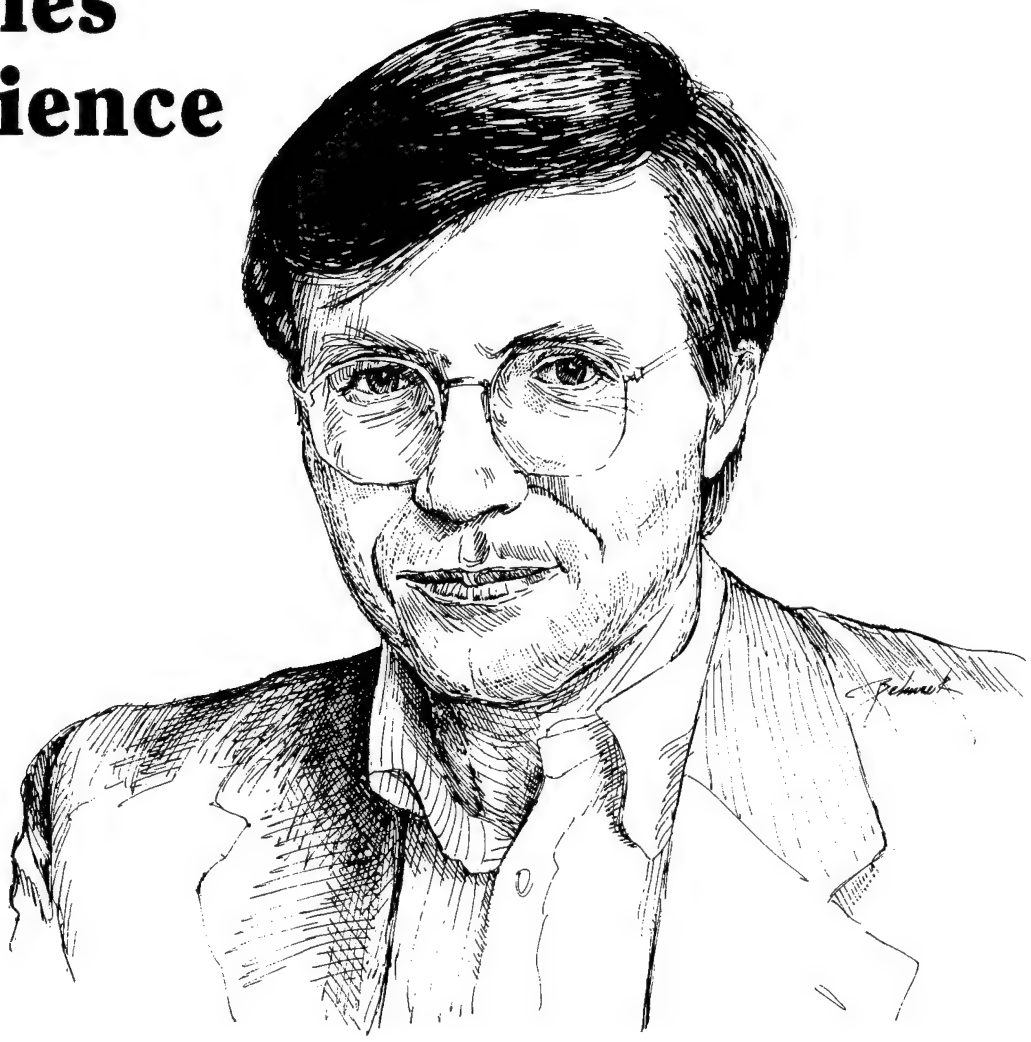
29. Arbib, M.A., & Dominey, P.F. "Modeling the Roles of Basal Ganglia in Timing and Sequencing Saccadic Eye Movements", in Houk, J.C., Davis, J.L., & Beiser, D.G. (Eds.), *Models of Information Processing in the Basal Ganglia*. (Cambridge, MA: MIT) (1995).

30. McHaffie, J.G., Norita, M., Dunning, D.D., and Stein, Barry E. "Corticotectal Relationships: Direct and "Indirect" Corticotectal," *Progress in Brain Research*, Vol. 95, Elsevier Science Publishers (1993).

31. Aronson, R.M. "Electronic Support Measures and Bistatic Radar Sensor Integration", *Proceedings of the 4th Nat'l. Symposium on Sensor Fusion*, April 2-4, 87 (1991).

32. Wright, F.L. "The Fusion of Multisensor Data," *Signal*, October, 39-43 (1980).

Profiles in Science



Dr. Terrence J. Sejnowski

Dr. Terrence J. Sejnowski is an Investigator with the Howard Hughes Medical Institute and a Professor at the Salk Institute for Biological Studies where he directs the Computational Neurobiology Laboratory. In 1984 while on the faculty of the Johns Hopkins University, he received a Presidential Young Investigator Award. He founded the journal *Neural Computation* in 1988 and with Patricia Churchland, wrote *The Computational Brain*, a primer on computational neuroscience.

Dr. Sejnowski has been a pioneer in computational neuroscience with primary research interests in vision, learning and memory, and biophysical mechanisms for neural computation. As a Principal Investigator of the Office of Naval Research, he was pursued research into novel forms of synaptic plasticity (neural network models for the distributed representation of space and the segmentation of moving objects in the visual cortex) and methods for assessing vigilance in sonar operators.

He recently introduced a nonlinear generalization of principal components analysis called independent components analysis that has widespread applications to underwater communication, echo cancellation, image compression, and the analysis of electroencephalographic recordings. The long-range goal of Dr. Sejnowski's research is to uncover principles that link brain to behavior using computational models.

A Digital VLSI Architecture for Neural Network Emulation Pattern Recognition, and Image Processing*

Dan Hammerstrom, Adaptive Solutions, Inc.

Introduction

As the other articles of this journal show, the neural network model has significant advantages over traditional models for certain applications. It has also expanded our understanding of biological neural networks by providing a theoretical foundation and a set of functional models.

Neural network simulation remains a computationally intensive activity, however. The underlying computations – generally multiply-accumulates – are simple but numerous. For example, in a simple artificial neural network (ANN) model, most nodes are connected to most other nodes, leading to $O(n^2)$ *connections*[1]. A network with 100,000 nodes, modest by biological standards, would therefore have about 10 billion connections, in the simplest models, with a multiply-accumulate operation needed for each connection. If a state-of-the-art workstation can simulate roughly 10 million connections per second, then one pass through the network takes 1,000 seconds (about 20 minutes). This data rate is much too slow for real-time process control or speech recognition, which must update several times a second. Clearly, we have a problem.

This performance bottleneck is worse if each connection requires more complex computations, for instance for incremental learning algorithms or for more realistic biological

* This paper is adapted from a chapter, "A Digital VLSI Architecture for Real-World Applications," Dan Hammerstrom, in *An Introduction to Neural and Electronic Networks - Second Edition*, pp. 335-358, Eds S.F. Zornetzer, J.L. Davis, C. Lau, and T. McKenna, Academic Press, 1995.

[1] The "order of" O(F(n)) notation means that the quantity represented by O is approximate for the function F within a multiplication or division of n by a constant.

simulations. Eliminating this computational barrier has lead to much research into building custom Very Large Scale Integration (VLSI) silicon chips optimized for ANNs. Such chips might perform ANN simulations hundreds to thousands of times faster than workstations or personal computers – for about the same cost.

The research into VLSI chips for neural network and pattern recognition applications is based on the premise that optimizing the chip architecture to the computational characteristics of the problem lets the designer create a silicon device offering a big improvement in performance/cost or "operations per dollar." In silicon design, the cost of a chip is primarily determined by its two-dimensional area. Smaller chips are cheaper chips. Within a chip, the cost of an operation is roughly determined by the silicon area needed to implement it. Furthermore, speed and cost usually have an inverse relationship: faster chips are generally bigger chips.

The silicon designer's goal is to increase the number of operations per unit area of silicon, called *functional density*, in turn increasing operations per dollar. An advantage of ANNs is that they employ simple, low-precision operations requiring little silicon area. As a result, chips designed for ANN emulation can have a higher functional density than traditional chips such as microprocessors. The motive for developing specialized chips, whether analog or digital, is this potential to improve performance, reduce cost, or both.

The designer of ANN silicon faces many other choices and trade-offs. One of the most important is flexibility versus speed. At the "specialized" end of the flexibility spectrum, the designer gives up versatility for speed to make a fast chip dedicated to one task. At the "general purpose" end, the sacrifice is reversed, yielding a slower, but programmable device. The choice is difficult because both traits are desirable. Real-world neural network applications ultimately need chips across the entire spectrum.

This paper reviews one such architecture, CNAPS[2] (Connected Network of Adaptive ProcessorS), developed by Adaptive Solutions, Inc. This architecture was designed for ANN simulation, image processing, and pattern recognition. To be useful in these related contexts, it occupies a point near the "general purpose" end of the flexibility spectrum. We believe that, for its intended markets, the CNAPS architecture has the right combination of speed and flexibility.

This paper is divided into two major sections, each framed in terms of the capabilities needed in the CNAPS computer's target markets. The first section presents an overview of the CNAPS architecture and offers a rationale for its major design decisions. It also summarizes the architecture's limitations and describes aspects that, in hindsight, its designers might have changed. The section ends with a brief discussion of the CNAPS program development software.

The second section briefly reviews applications developed for CNAPS at this writing[3]. The applications discussed are simple image processing, automatic target recognition, a simulation of the Lynch/Granger Pyriform Model, Kanji OCR, Adobe PhotoShop acceleration, and medical image processing.

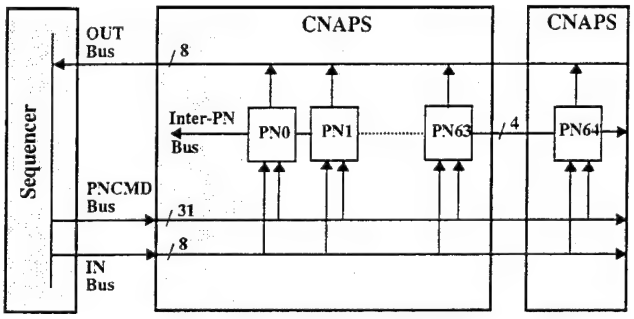
The CNAPS Architecture

The CNAPS architecture consists of an array of processors controlled by a sequencer, both implemented in a chip set developed by Adaptive Solutions, Inc. The sequencer is a one-chip device, called the CNAPS Sequencer Chip (CSC). The processor array is also a one-chip device, available with either 64 or 16 processors per chip (the CNAPS-1064 or CNAPS-1016). The CSC can control up to eight 1064s or 1016s, which act like one larger device.

These chips usually sit on a printed circuit board that plugs into a host computer, also called the control processor (CP). The CNAPS board acts as a coprocessor within the host. Under the coprocessor model, the host sends data and programs to the board, which runs until done, then interrupts the host to indicate completion. This style of operation is called "run to completion semantics." Another possible model is to use the

Figure 1.

The basic CNAPS Architecture. CNAPS is a single instruction multiple data (SIMD) architecture that uses broadcast input, one-dimensional inter-processor communication, and a single, shared output bus.



[2] Trademark Adaptive Solutions, Inc.

[3] Because ANNs are becoming a key technology, many customers consider their use of ANNs to be proprietary information. Many applications are not yet public knowledge.

CNAPS board as a stand-alone device to process data continuously.

The CNAPS Architecture

Basic Structure

CNAPS is a single instruction, multiple data stream (SIMD) architecture. SIMD computers have one instruction sequencing/control unit and many processor nodes (PNs). In CNAPS, the PNs are connected in a one-dimensional array (Figure 1) where each PN can "talk" only to its right or left neighbors. The sequencer broadcasts each instruction plus input data to all PNs, which execute the same instruction at each clock. The PNs transmit output data to the sequencer, with several arbitration modes controlling access to the output bus.

As Figure 2 suggests, each PN has a local memory[4], a multiplier, an adder/subtractor, a shifter/logic unit, a register file,[5] and a memory addressing unit. The entire PN uses fixed-point, two's complement arithmetic, and the precision is 16 bits with some exceptions. The PN memory can handle 8- or 16-bit reads or writes. The multiplier produces a 24-bit output; an 8x16 or 8x8 multiply takes one clock, and a 16x16 multiply takes two clocks. The adder can switch between 16- or 32-bit modes. The input and output buses are 8 bits wide, and a 16-bit word can be assembled (or disassembled) from two bytes in two clocks.

A PN has several additional features, [7] and [6], including a function that finds the PN with the largest or smallest values (useful for winner-take-all and best-match operations), various precision and memory control features, and OutBus arbitration. These features are too detailed to discuss fully here.

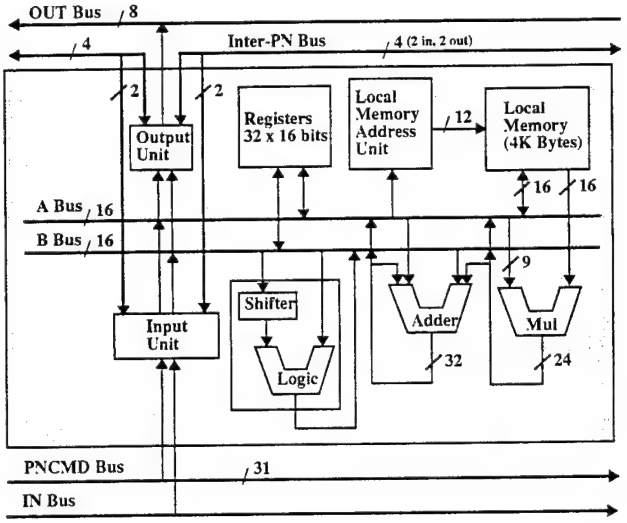
The CSC sequencer (Figure 3) performs program sequencing for the PN array and has private access to a program memory. The CSC also performs I/O processing for the array, writing input data to the array and reading output data from it. To move data to and from CP memory, the CSC has a 32-bit bus, called the AdaptBus, on the CP side. The CSC also has a direct input port and a direct output port used to connect the CSC directly to I/O devices for higher-bandwidth data movement.

Neural Network Example

The CNAPS architecture can execute many ANN and non-ANN algorithms. Many SIMD techniques are the same in both contexts, so an ANN can serve as a general example of mapping an algorithm to the array. Specifically, the example shows how the PN array simulates a layer in an ANN.

Figure 2.

The internal structure of a CNAPS processor node (PN). Each PN has its own storage and arithmetic capabilities. Storage consists of 4,096 bytes. Arithmetic operations include multiply, accumulate, logic, and shift. All units are interconnected by two buses.



Start by assuming a two-layered network (Figure 5) where – for simplicity – each node in each layer maps to one PN. PN_i thus simulates the node n_{ij}, where i is the node index in the layer and j is the layer index. Layers are simulated in a time-multiplexed manner. All layer 1 nodes thus execute as a block, then all layer 2 nodes, etc. Finally assume that layer 1 has already calculated its various $n_{i,1}$ outputs.

The goal at this point is to calculate the outputs for layer 2. To achieve this, all layer 1 PNs simultaneously load their output values into a special output buffer and begin arbitrating for the output bus. In this case, the arbitration mode lets each PN transmit its output in sequence. In one clock, the content of PN_0's buffer is placed on the output bus and goes through the sequencer[6] to the input bus. From the input bus, the value is broadcast to all PNs (this out-to-in loopback feature is a key to implementing layered structures efficiently). Each PN then multiplies node $n_{0,1}$'s output with a locally stored weight, $w_{0,1}$.

On the next clock, node $n_{1,1}$'s output is broadcast to all PNs, and so on for the remaining layer 1 output values. After N clocks, all outputs have been broadcast, and the inner product computation is complete. All PNs then use the accu-

[4] Currently 4KB per PN.
[5] Currently 32, 16-bit registers.
[6] This operation actually takes several clocks and must be pipelined. These details are eliminated here for clarity.

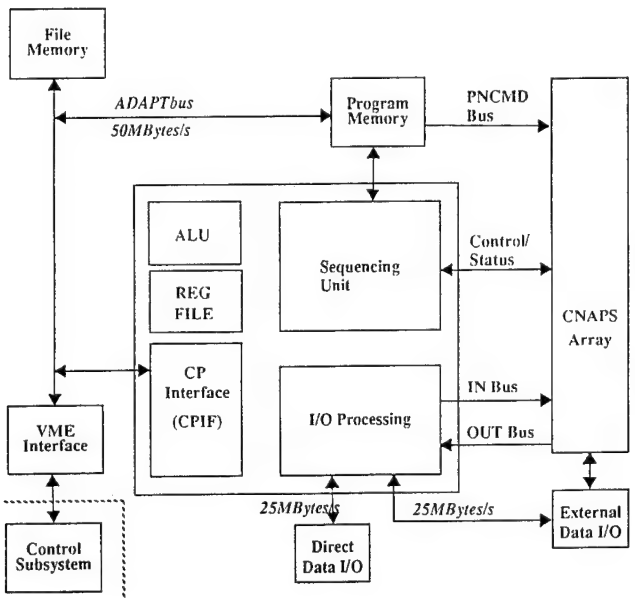
mulated value's upper 8 bits to look up an 8-bit non-linear output value in a 256 item table stored in each PN's local memory. This process – calculating a weighted sum, then passing it through a function stored in a table – is performed for each output on each layer. The last layer transmits its output values through the CSC to an output buffer in the CP memory.

The multiply-accumulate pipeline can compute a connection in each clock. The example network has four nodes and uses only four clocks for its 16 connections. For even greater efficiency, other operations can be performed in the same clock as the multiply-accumulate. The separate memory address unit, for instance, can compute the next weight's address at the same time as the connection computation. And the local memory allows the weight to be fetched without delay.

An array of 256 PNs can compute 2562 = 65536 connections in 256 clocks. At a 25 MHz clock frequency, this equals 6.4 billion connections per second (back-propagation feed-forward) and over 1 billion connection updates per second (back-propagation learning). An array of 64 PNs (one CNAPS-1064 chip), for example, can store and train the entire NetTalk [18] network in about 7 seconds.

Figure 3.

The CNAPS sequencer chip (CSC) internal structure. The CSC accesses an external program store, which contains both CSC and CNAPS PN array instructions. PN array instructions are broadcast to all PNs. CSC instructions control sequencing and all array input and output.



Physical Implementation

The CNAPS PN array has been implemented in two chips, one with 64 PNs (the CNAPS-1064[4]) (Figure 5) and the other with 16 PNs (the CNAPS-1016). Both chips are implemented in a 0.8 micron CMOS process. The 64 PN chip is a full custom design and is about 26 millimeters on a side and has over 14 million transistors, making it one of the largest processor chips ever made. The simple computational model makes possible a small, simple PN, in turn permitting the use of redundancy to improve semiconductor yield for such a device.

The CSC is implemented using a gate array technology, using a 100,000 gate die and is about 10 millimeters on a side.

The next section reviews the various design decisions and the reasons for making them. Some of the features described are unique to CNAPS; others apply to any Digital Signal Processor chip.

Major Design Decisions

When designing the CNAPS architecture, a key question was where it should sit relative to other computing devices in cost and capabilities. In computer design, flexibility and performance are almost always inversely related. We wanted CNAPS to be flexible enough to run a broad family of ANN algorithms as well as other related pattern recognition and preprocessing algorithms. Yet we wanted it to have much higher performance than state-of-the-art workstations and – at the same time – lower cost for its functions.

Figure 6 shows where we are targeting CNAPS. The vertical dimension plots each architecture by its flexibility. Flexibility is difficult to quantify, since it involves not only the range of algorithms that an architecture can execute, but also the complexity of the problems it can solve. (Greater complexity typically requires a larger range of operations.) As a result, this graph is subjective and provided only as an illustration.

The horizontal dimension plots each architecture by its performance/cost – or operations per dollar. The values are expressed in a log scale due to the orders-of-magnitude difference between traditional microprocessors at the low end and highly-custom, analog chips at the high end. Note the *technology barrier*, defined by practical limits of semiconductor technology. No one can build past the barrier, since you can do only so much with a transistor; you can put only so many of them on a chip; and you can run them only so fast.

For ANN emulation, we wanted to place the CNAPS architecture in the middle, between the specialized analog chips and the general purpose microprocessors. We wanted it to be programmable enough to solve many real-world problems – and yet have a performance/cost about 100 times faster than the highest performance RISC processors.

In determining the degree of function required, we must solve all or most of a targeted problem. This need results from

Amdahl's Law, which states that system performance depends mainly on the slowest component. This law can be formalized as follows:

$$S = \frac{1}{(op_f * s_f) + (op_h * s_h)}$$

where S is the total system speed-up, op_f is the fraction of total operations in the part of the computation run on the fast chip, s_f is the speed-up the chip provides, op_h is the fraction of total operations run on a host computer without acceleration. Hence as op_f or s_f get large, S approaches $1/op_h$. Unfortunately, op_f needs to be close to one before any real system-level improvement occurs, as shown in the following example.

Suppose there are two such support chips to choose from: the first can run 80% of the computation with 20x improvement on that 80%; the second can run only 20%, but runs that 20% 1000x faster. By Amdahl's law, the first chip speeds up the system by over 400%, while the second – and seemingly faster – chip speeds up the system by only 20%. Amdahl tells us, therefore, that flexibility is often better than raw performance, especially if that performance results from limiting the range of operations performed by the device.

Digital

Much effort has been dedicated to building analog VLSI chips for ANNs. Analog chips have great appeal, partly because they follow biological models more closely than digital chips. Analog chips also can achieve higher functional density. Excellent papers reporting research in this area include [12], [1], [5], [10], and [2]. And see Morgan, [15], for a good summary of digital neural network emulation.

Analog ANN implementations have been primarily academic or industrial research projects, however. Only a few have found their way into the real world as commercial products: getting an analog device to work in a laboratory is one thing; making it work over a wide range of voltages, temperatures, and user capabilities is another. In general, analog chips require much more stringent operating conditions than digital chips. They are also more difficult to design and, after implementation, less flexible.

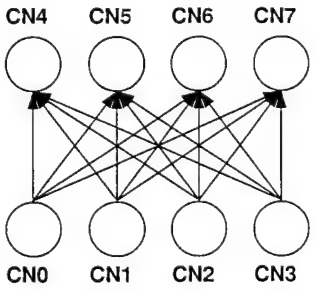
The semiconductor industry is heavily oriented toward digital chips. Analog chips represent only a minor part of the total output, reinforcing their secondary position. There are, of course, successful analog parts, and there always will be, since some applications require analog's higher functional density to achieve their cost and performance constraints, and those applications can tolerate analog's limited flexibility. Likewise, there will be successful products using analog ANN chips. Analog parts will probably be used in simple applications or as a part of larger system in more complex applications, however.

This prediction follows primarily from their limited flexibility. Analog chips typically implement one algorithm hardwired into the chip. A hardwired algorithm is fine if truly stable. The field of ANN applications is still new, however, so most complex implementations are still actively evolving – even at the algorithm level. An analog device cannot easily follow such changes. A digital, programmable device can change algorithms by changing software.

Our major goal was to produce a commercial product that would be flexible enough and provide sufficient precision to cover a broad range of complex problems. This goal dictated a digital design, since digital could offer better precision and much more flexibility than a typical CMOS analog implementation. Digital also offered excellent performance and all the advantages of a standardized technology.

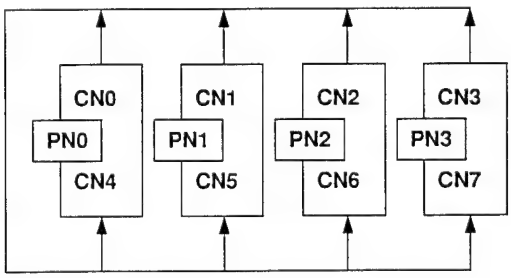
Figure 4.

A simple two-layered neural network. In this example each PN emulates two network nodes. PNs emulate the first layer, computing one connection each clock. They then sequentially place node output on the OutBus while emulating, in parallel, the second layer.



Broadcast by PN0 of CN0's output to CN4, 5, 6, 7 takes 1 clock

N^2 connections in N clocks



Limited, Fixed-Point Precision

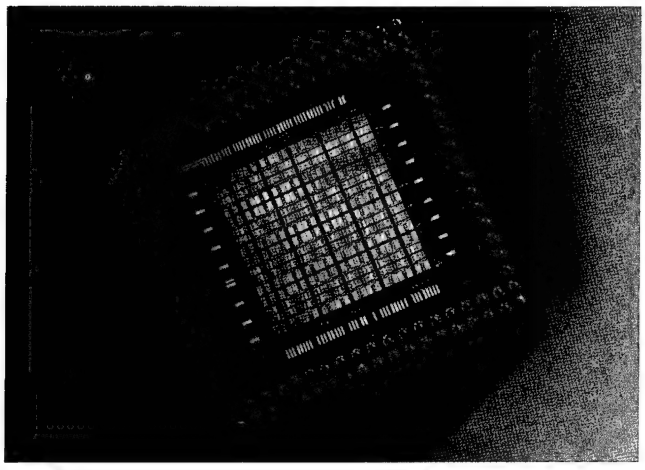
In both analog and digital domains, an important decision is choosing the arithmetic precision required. In analog, precision affects design complexity and the amount of compensation circuitry required. In digital, it effects the number of wires available as well as the size and complexity of memory, buses, and arithmetic units. Precision also affects the power dissipation.

In the digital domain, a related decision involves floating-point versus fixed-point representation. Floating-point numbers (Figure 7) consist of an exponent (usually 8 bits representing base 2 or base 16) and a mantissa (usually 24 bits). The exponent is set so that the mantissa is always normalized – that is, the most significant one is in the most significant position. Adding two floating-point numbers involves shifting at least one of the operands to get the same exponent. Multiplying two floating-point numbers involves separate arithmetic on both exponents and mantissas. Both operations require post-operation normalization shifting after the arithmetic operations.

Floating point has several advantages. The primary advantage is dynamic range, which results from the separate exponent. Another is precision, due to the 24-bit mantissas. The disadvantage to floating point is its cost in silicon area. Much circuitry is required to keep track of both exponents and mantissas and perform pre- and post-operation shifting of the mantissa. This circuitry is particularly complicated if high speed is required.

Figure 5.

The CNAPS PN array chip. There are 64PNs with memory on each die. The PN array chip is one of the largest processor chips ever made. It consists of 14 million transistors and is over 26 millimeters on a side. PN redundancy, there are 16 spare PNs, is used to guarantee high yields.



Fixed-point numbers consist of a numeral (usually 16 to 32 bits) and a radix point (in base two, the binary point). In fixed point, the programmer chooses the position of the radix point. This position is typically fixed for the calculation, although it is possible to change the radix point under software control by explicitly shifting operands. For many applications needing only limited dynamic range and precision, fixed point is sufficient. It is also much cheaper than floating point because it requires less silicon area.

After choosing a digital signal representation for CNAPS, the next question was how to represent the numbers. Biological neurons are known to use relatively low precision and to have a limited dynamic range. These characteristics strongly suggest that a digital computer for emulating ANN structures should be able to employ limited precision fixed-point arithmetic. This conjecture in turn suggests an opportunity to significantly simplify the arithmetic units and to provide greater computational density. Fixed-point arithmetic also places the design near the desired point on the flexibility versus performance/cost curve (Figure 6).

To confirm the supposition that fixed-point is adequate, we performed extensive simulations. We found that for the target applications, 8- or 16-bit fixed-point precision was sufficient[3]. Other researchers have since reached the same conclusion, [9] and [19]. In keeping with experimental results, we used a general 16-bit resolution inside the PN. One exception was using a 32-bit adder to provide additional head room for repeated multiply-accumulates. Another was using 8-bit input and output data buses, since most computations involve 8-bit data and 8- or 16-bit weights, and since busing external to the PN is expensive in silicon and board area. Using 16 bits for the buses internal to the PN did not add that much extra area.

SIMD

The next major decision was how to control the PNs. A computer can have one or more instruction streams and one or more data streams. Most computers are single instruction, single data (SISD) computers. These have one control unit and one processor unit, usually combined on one chip (a microprocessor). The control unit fetches instructions from program memory and decodes them. It then sends data operations such as add, subtract, or multiply to the processing unit. Sequencing operations, such as branch, are executed by the control unit itself. SISD computers are serial, not parallel.

Two major families of parallel computer architectures have evolved: multiple instruction, multiple data (MIMD) and single instruction, multiple data (SIMD). MIMD computers have many processing units, each of which has its own control unit. Each control/processing unit can operate in parallel, executing many instructions at once. Since the processors operate independently, MIMD is the most powerful and flexible parallel architecture. The independent, asynchronous

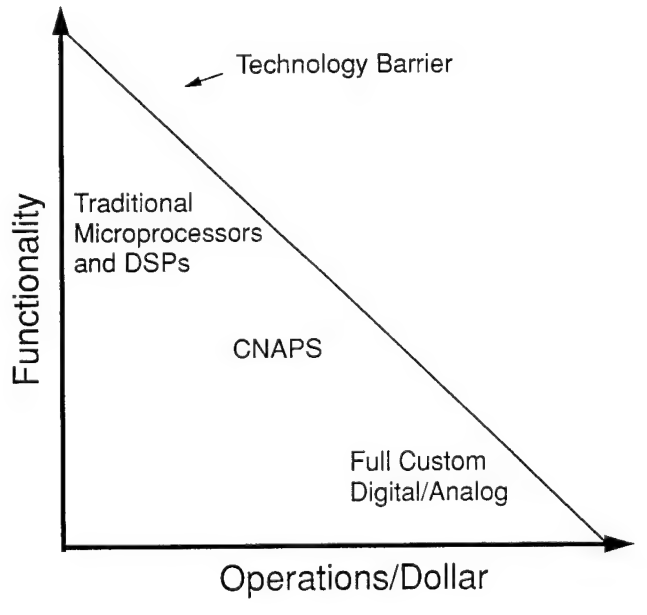
processors also make MIMD the most difficult to use, requiring complex processor synchronization.

SIMD computers have many processors but only one instruction stream. All processors receive the same instruction at the same time, but each acts on its own slice of the data. SIMD computers thus have an array of processors and can perform an operation on a block of data in one step. SIMD computing is often called "data parallel" computing, since it applies one control thread to multiple, local data elements, executing one instruction at each clock.

SIMD computation is perfect for vector and matrix arithmetic. Due to Amdahl's law, however, SIMD is cost effective only if most operations are matrix or vector operations. For general-purpose computing, that is not the case. Consequently, SIMD machines are poor general-purpose computers and rarer than SISD or even MIMD computers. Our target domain is not general-purpose computing, however. For ANNs and other image and pattern recognition, and signal processing algorithms, the dominant calculations are vector or matrix operations. SIMD fits this domain perfectly.

Figure 6.

Though subjective, this graph gives a rough indication of the CNAPS market positioning. The vertical dimension measures the range of functionality of an architecture; the horizontal dimension, the performance/cost in operations per dollar. The philosophy behind CNAPS is that by restricting functionality to pattern recognition, image processing, and neural network emulation, a larger performance/cost is possible than with traditional machines (parallel or sequential).



SIMD is a good choice for practical reasons too. One advantage is cost: SIMD is much cheaper than MIMD, since there is only one control unit for the entire array of processors. Another is that SIMD is easier to program than MIMD, since all processors do the same thing at the same time. Likewise, it is easier to develop computer languages for SIMD, since it is relatively easy to create parallel data structures where the data are operated on simultaneously. Figure 8 shows a simple CNAPS-C program that multiplies a vector times a matrix. Normally, vector matrix multiply takes n2 operations. By placing each column of the matrix on each PN, it takes n operations on n processors.

In sum, SIMD was better than MIMD for CNAPS because it fit the problem domain, was much more economical, and easier to program.

Broadcast Interconnect

The next decision concerned how to interconnect the PNs for data transfer both within the array and outside it. Computer architects have developed several interconnect structures for connecting processors in multiprocessor systems. Since CNAPS is a SIMD machine, we were interested only in synchronous structures.

The two families of interconnect are *local* and *global*. Local interconnect attaches only neighboring PNs. The most common local scheme is NEWS (North-East-West-South – Figure 9). In NEWS, the PNs are laid out in a two-dimensional array, and each PN is connected to its four nearest neighbors. A one-dimensional variation connects each PN only to its left and right neighbors.

Global interconnect permits any PN to talk to any other PN, not just to its immediate neighbors. There are several possible configurations with different levels of performance/cost. At one end of the scale, *cross-bar* interconnect is versatile since it permits random point-to-point communications, but expensive (the cost is $O(n^2)$, where n is the number of PNs). At the other end, *broadcast* interconnect is cheaper but less flexible. Here, one bus connects all PNs, so any one PN can talk to any other (or set of others) in one clock. On the other hand, it takes n clocks for all PNs to have a turn. The cost is O(1). In between crossbar and broadcast are other configurations that can emulate a crossbar in O(log n) clocks and have cost O(m log n).

Choosing an interconnect structure interacted with other design choices. We reached a crossroads by deciding against using a systolic computing style, where operands, intermediate results, or both flow down a row of PNs only using local interconnect. Systolic arrays are harder to program. They are also occasionally inefficient due to the clocks needed to fill or empty the pipeline – peak efficiency occurs only when all PNs see all operands. Choosing a systolic array would have permitted us to use local interconnect, saving cost. Deciding against it forced us to provide some form of global interconnect.

Choosing "global" leads to the next choice: what type? The basic computations in our target applications require "one-to-many" or "many-to-many" communication almost exclusively. We therefore decided to use a broadcast bus, which uses only one clock for one-to-many communication. In the many-to-many case, n PNs can talk to all n PNs in n clocks. Broadcast interconnect thus allows n^2 connections in n clocks. Such $O(n^2)$ total connectivity occurs often in ANN models. An example is a back-propagation network in which all nodes in one layer connect to all nodes in the next.

Another advantage is that broadcast interconnection is synchronous and fits the synchronous SIMD structure quite well. We were able to use a "slotted" protocol, where each connection occurs at a known time on the bus. Since the time is known, there is no need to send an address with each data element, saving wires, clocks, or both. Also, the weight address unit can "remember" the slot number and use it to address the weight associated with the connection.

A single broadcast bus is simple, economical to implement, and efficient for the application domain. In fact, if every PN always communicates with every other PN, then broadcast offers the best possible performance/cost.

Broadcast interconnection does have some drawbacks. One problem is its inefficiency for some point-to-point communication patterns, where one PN talks with another PN anywhere in the array. An example of such a pattern is the "perfect shuffle" used by the fast Fourier transform (FFT) (Figure 10). This pattern takes n clocks on the CNAPS broadcast bus and is too slow to be effective. Consequently, CNAPS implements the compute-intensive, discrete Fourier transform (DFT) instead of the communication-intensive FFT. The DFT requires $O(n^2)$ operations; the FFT, O(n log n). If $n \approx p$, where p is the number of PNs, then CNAPS can perform a DFT in O(n) clocks, however. If $\log n \approx p$, then performance can approach the O(n log n) of a sequential processor.

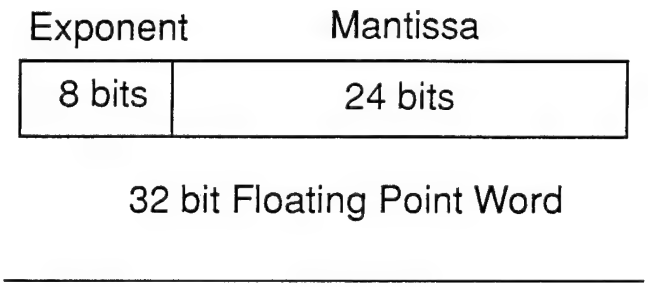
Another problem involves computation localized in a portion of an input vector, where each PN operates on a different (possibly overlapping) subset of the elements. Here, all PNs must wait for all inputs to be broadcast before any computation can begin. A common example of this situation is the limited receptive field structure, often found in image classification and character recognition networks. The convolution operation, also common in image processing, uses similar localized computation. The convolution can proceed rapidly after some portion of the image has been input into each PN, since each PN operates independently on its subset of the image.

When these subfields overlap (as in convolution), a PN must communicate with its neighbors. To improve performance for such cases, we added a one-dimensional inter-PN pathway, connecting each PN to its right and left neighbors. (One dimension was chosen over two to allow processor redundancy, discussed further below.) The CNAPS array therefore has both global (broadcast) and local (inter-PN) interconnection. An example of using the inter-PN pathway might be image processing, where a column of each image is allocated to each PN. The inter-PN pathway permits efficient communication between columns, and, consequently, efficient computation of most image processing algorithms.

A final problem is sparse random interconnect, where each node connects to some random subset of other nodes. Broadcast is, from the viewpoint of the *connected* PNs, efficient in this case. Nonetheless, when a sparse connectivity is used with a slotted protocol, many PNs are idle, since they lack weights connected to most inputs and cannot use most of the data being broadcast. Sparse interconnect affects all aspects of the architecture, not just data communication. To improve efficiency for sparsely-connected networks, the CNAPS PN offers a special memory technique called *virtual zero*, which saves memory locations, which would otherwise be filled with zeros, by not loading zeros into memory for unused connections. The Virtual Zero technique does not help the idle PN problem, however. Full efficiency with sparse interconnect requires a much more complex architecture, including more individualized control per PN, more complex memory referencing capabilities, etc. and its discussion is beyond the scope of this paper.

Figure 7.

A floating point number. A single precision, IEEE compatible floating point configuration is shown. The high order 8 bits constitute the exponent; the remaining 24 bits, the mantissa or "fractional" part. Floating point numbers are usually normalized so that the mantissa has a 1 in the most significant position.



[7] For most implementations the bit rate per pin is roughly equal to the clock rate, which can vary anywhere from 25 to 200 MHz. There are some special interface protocols which now allow up to 500 Megabits per second per pin, but power dissipation limits how many bits can be sent off chip at those frequencies.

Figure 8.

A CNAPS-C program to do a simple vector-matrix multiply. The "data-parallel" programming is evident here. Within the loop, it is assumed, because of the domain declaration, that there are multiple copies of each matrix element, one on each PN. The program takes N loop iterations, which would require N^2 on a sequential machine.

```
# define N 20
# define K 30
typedef scaled 8 8 arithType;
domain Krows
      {arithType sourceMatrix[N];
      arithType resultVector;} dimK[K];


main()
{ int n;
      [domain dimK].{
      resultVector = 0;
      for (n=0; n < N; n++)
      resultVector += sourceMatrix[n] * getchar();
      }
}
```

On-Chip Memory

One of the most difficult decisions was whether to place the local memory on-chip (inside the PN) or off-chip. Both approaches have advantages and drawbacks. It was a complex decision with no obvious right answer and little opportunity for compromise.

The major advantage of off-chip memory is that it allows essentially unlimited memory per PN. Placing memory inside the PN, in contrast, limits the available memory because memory takes significant silicon area. Increasing PN size also limits the number of PNs. Another advantage to off-chip memory is that it allows the use of relatively low-cost, commercial memory chips. On-chip memory, in contrast, increases the cost per bit – even if the memory employs a commercial memory cell.

The major advantage of on-chip memory is that it allows much higher bandwidth for memory access. To see bandwidth as a crucial factor, consider the following analysis. Recall that each PN has its own data arithmetic units, so each PN requires a unique memory data stream. The CNAPS-1064 has 64 PNs, each potentially requiring up to two bytes per clock. At 25 MHz, that is 25M * 64 * 2 = 3.2 billion bytes per second. Attaining 3.2 billion bytes per second from off-chip memory is difficult and expensive due to limits on the number of pins per chip and the data rate per pin. An option would be to reduce the number of PNs per chip, eroding the benefit of maximum parallelism.

Another advantage to on-chip memory is that each PN can address different locations in memory each clock. Systems with off-chip memory, in contrast, typically require all PNs to address the same location for each memory reference to reduce the number of external output pins. With a shared address only a single set of address pins is required for an entire PN array. Allowing each PN to have unique memory addresses, requires a set of address pins for each PN, which is expensive.. Yet having each PN address its own local memory improves versatility and speed, since table lookup, string operations, and other kinds of "indirect" reference are possible.

Yet another advantage is that the total system is simpler. On-chip memory makes it possible to create a complete system with little more than one sequencer chip, one PN array chip, and some external RAM or ROM for the CSC program. (Program memory needs less bandwidth than PN memory because SIMD machines access it serially, one instruction per clock.)

It is possible to place a cache in each PN, then use off-chip memory as a backing store, which attempts to gain the benefits of both on-chip and off-chip memory by using aspects of both designs. Our simulations on this point verified what most people who work in ANNs already suspected: caching is ineffective for ANNs due to the non-locality of the memory references streams. Caches are effective if the processor repeatedly accesses a small set of memory locations, called a *working set*. ANNs rarely exhibit that kind of behavior; instead, they reference long, sequential vector arrays (generally weights).

Separate PN memory addressing also reduces the benefit of caching. Unless all PNs refer to the same address, some PNs can have a cache miss and others not. If the probability of a cache miss is 10% per PN, then a 256 PN array will most likely have a cache miss every clock. But due to the synchronous

[8] CNAPS-C is a data parallel version of the standard C language.

SIMD control, all PNs must wait for the one or more PNs that miss the cache. This behavior renders the cache useless. A MIMD structure overcomes the problem – but increases system complexity and cost.

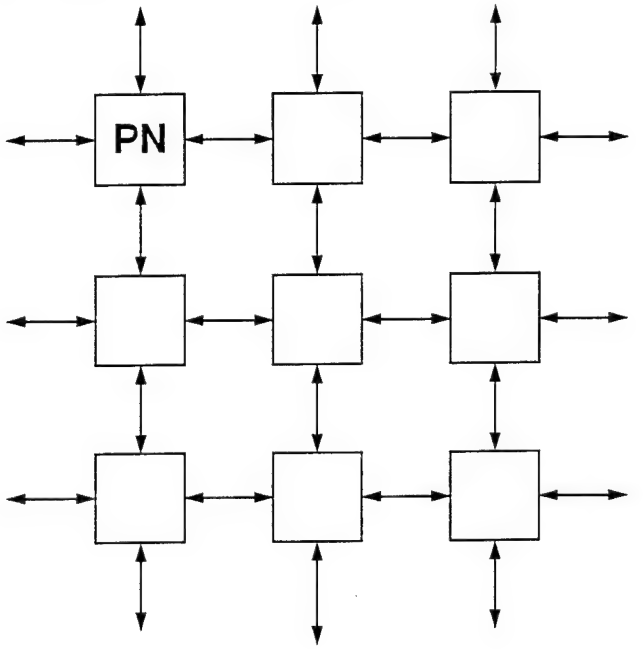
As this discussion suggests, local PN memory is a complex topic with no easy answers. Primarily due to bandwidth needs and the access to a commercial density static RAM CMOS process, we decided to implement PN memory on chip, inside the PN. Each PN has 4 KB in the current 1064 and 1016 chips.

CNAPS is the only architecture for ANN applications we are aware of that uses on-chip memory. Several designs have been proposed that use off-chip memory. The CNS system being developed at Berkeley [21], for instance, restricts the number of PNs to 16 per chip. It also uses a special high-speed PN-to-memory bus to achieve the necessary bandwidth. Another system, developed by Ramacher and others at Siemens, [16] uses a special systolic pipeline that reduces the number of fetches required by forcing each memory fetch to be used several times. This organization is efficient at doing inner products, but has restricted flexibility. HNC has also created a SIMD array called the SNAP [14]. It uses floating-point arithmetic, reducing the number of PNs on a chip to only four – in turn reducing the bandwidth requirements.

The major problem with on-chip memory is its limited memory capacity. While this limitation does restrict CNAPS applications, it has not been a major problem. With early applications, the performance/cost advantages of on-chip memory have been more important than the memory capacity limits.

Figure 9.

A two-dimentional PN layout. This configuration is often called, a "NEWS" network, since each PN connects to its north, east, west, and south neighbor. These networks provide more flexible intercommunication than a one-dimentional network, but are difficult to make work when redundant PNs are used.



Redundancy for Yield Improvement

During the manufacture of integrated circuits, small defects and other anomalies occur, causing some circuits to malfunction. These defects have a more-or-less random distribution on a silicon wafer. The larger the chip, the greater the probability that at least one defect will occur there during manufacturing. The number of good chips per wafer is called the *yield*. As chips get larger, fewer chips fit on a wafer, and more have defects, therefore, yield drops off rapidly with size. Since wafer costs are fixed, cost per chip is directly related to good chips per wafer. The result is that bigger chips cost more. On the other hand, bigger chips do more, and their ability to fit more function into a smaller system makes big chips worth more. Semiconductor engineers are constantly pushing the limits to maximize both function and yield at the same time.

One way to build larger chips and maximize yield is to use *redundancy*, where many copies of a circuit are built into the chip. After fabrication, defective circuits are switched out and replaced with a good copy. Memory designers have used redundancy for years: extra memory words are fabricated on the chip and substituted for defective words. With redundancy, some defects can be tolerated and still yield a fully functional chip.

One advantage of building ANN silicon is that each PN can be simple and small. In the CNAPS processor array chip, the PNs are small enough to be effective as "units of redundancy." By fabricating spare PNs, we can significantly improve yield and reduce the cost per PN. The 1064 has 80 PNs (in an 8x10 array), and the 1016 has 20 (4x5). Even with a relatively high defect density, the probability of at least 64 out of 80 (or 16 out of 20) PNs being fully functional is close to 1.0. CNAPS is the first commercial processor to make extensive use of such redundancy to reduce costs. Without redundancy, the processor array chips would have been smaller and less cost-effective. We estimate a CNAPS implementation

[9] To change algorithms, the CSC need only branch to a different section of a program.

using redundancy has about a two-times performance/cost advantage over one lacking redundancy.

Redundancy also influenced the decision to use limited-precision, fixed-point arithmetic. Our analyses showed that floating-point PNs would have been too large to leverage redundancy; hence, floating point would have been even more expensive than just the size difference (normally about a factor of four) indicates.

Redundancy also influenced the decision to use one-dimensional inter-PN interconnect. One-dimensional interconnect makes it relatively easy to implement PN redundancy, since any 64 of the 80 PNs can be used. Two-dimensional interconnect complicates redundancy and was not essential for our applications. We chose one-dimensional interconnect, since it was adequate for our applications and does not impact the PN redundancy mechanisms.

Limitations

In retrospect, we are satisfied with the decisions made in designing the CNAPS architecture. We have no regrets about the major decisions such as the choices of digital, SIMD, limited fixed point, broadcast interconnect, and on-chip memory.

The architecture does have a few minor bottlenecks that will be improved in future versions. For example, the 8-bit input/output buses should be 16-bit. In line with that, a true one-clock 16x16 multiply is needed, as well as better support for rounding. And future versions will have higher frequencies and more on-chip memory. A hardware based random number generator for each PN would also be useful for many ANN emulation tasks. Despite these few limitations, the architecture has been successfully applied to several applications with excellent performance.

Product Realization and Software

Adaptive Solutions has created a complete development software package for CNAPS. It includes a library of important ANN algorithms and a C compiler with a library of commonly used functions. Several board products are now available and sold to customers to use for ANN emulation, image and signal processing, and pattern recognition applications.

CNAPS Applications

This section reviews several CNAPS applications. Its focus is on ANN and non-ANN applications. Some applications mix ANN and non-ANN techniques. For example, an application could preprocess and enhance an image via standard imaging algorithms, then use an ANN classifier on segments of the image, keeping all data inside the CNAPS array for all operations

Back-Propagation

The most popular ANN algorithm is back-propagation (BP) [17]. Although requiring large computational resources during training, BP has several advantages that make it a valuable algorithm:

- BP is reasonably generic, meaning that one network model (emulation program) can be applied to a wide range of applications with little or no modification;
- its non-linear, multilayer architecture lets it solve complex problems;
- BP is relatively easy to use and understand; and
- several commercial software vendors have excellent BP implementations.

It is estimated that over 90% of the ANN applications in use today use BP or some variant of it. We therefore felt that it was important for CNAPS to execute BP efficiently. This section briefly discusses the general implementation of BP on CNAPS. For more detail, see McCartor [11].

Figure 10.

The intercommunication pattern of a fast Fourier transform (FFT) A butterfly intercommunication pattern for four nodes. This pattern is difficult to do in less than N clocks (where N is the number of nodes) with broadcast inter-communications.

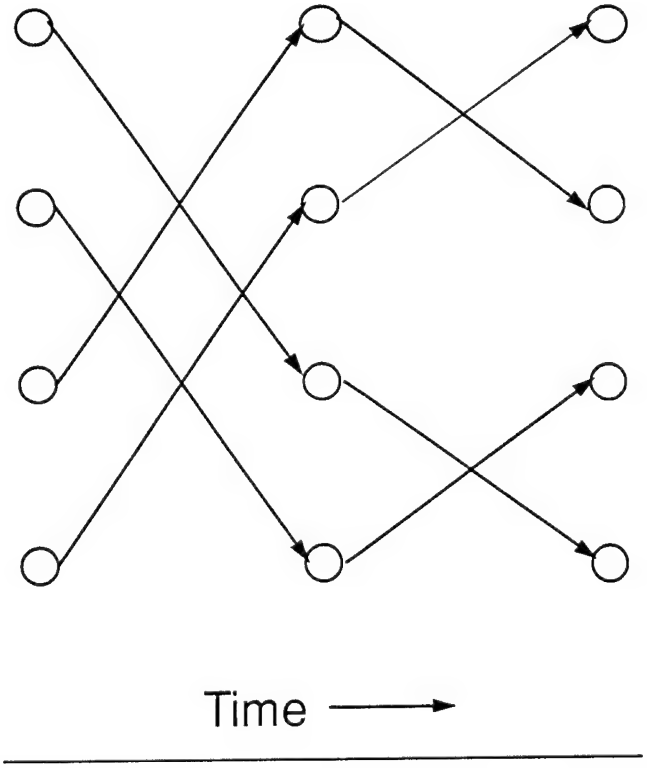
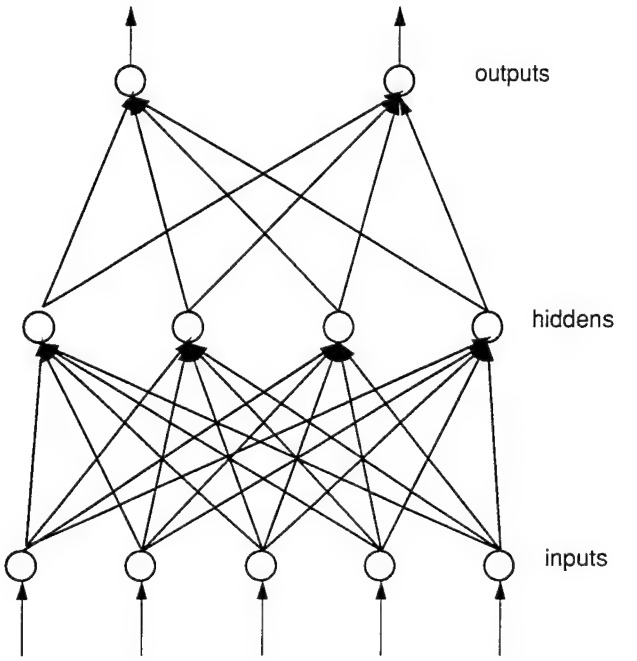


Figure 11.

A back-propagation network with five inputs, four hidden nodes and two output nodes.



There are two CNAPS implementations of BP, a single-precision version (BP16) and a double-precision version (BP32). BP16 uses unsigned 8-bit input and output values and signed 16-bit weights. The activation function is a traditional sigmoid, implemented by table lookup. BP32 uses signed 16-bit input and output values and signed 32-bit weights. The activation function is a hyperbolic tangent implemented by table lookup for the upper 8 bits and by linear extrapolation for the lower 8 bits. All values are fixed point. We have found that BP16 is sufficient for all classification problems. BP16 has also been sufficient for most curve fitting problems, such as function prediction, which have more stringent accuracy requirements. In those cases where BP16 does not have the accuracy of floating point, BP32 is as accurate as floating point in all cases studied so far. The rest of this section focuses on the BP16 algorithm. It does not discuss the techniques involved in dealing with limited precision on CNAPS.

BP has two phases. The first is feed-forward operation, where the network passes data without updating weights. The second is error back-propagation and weight update during training. Each phase will be discussed separately. This discussion assumes that the reader already has a working understanding of BP.

Back-Propagation: Feedforward Phase

Assume a simple CNAPS system with four PNs and a BP network with five inputs, four hidden nodes, and two output nodes (34 total connections, counting a separate bias connection for each node) (Figure 11). Allocate nodes 0 and 4 to PN0, nodes 1 and 5 to PN1, node 2 to PN2, and node 3 to PN3. When a node is allocated to a PN, the local memory of that PN is loaded with the weight values for each of the node's connections and with the lookup table for the sigmoid function. If learning is to be performed, then each connection requires a two-byte weight plus two bytes to accumulate the weight deltas, and a 2-byte transpose weight (discussed below). This network then requires 204 bytes for connection information and 256 bytes for the lookup table. Using momentum – ignored here for simplicity – would require more bytes per connection.

Each input vector contains five elements. To start the emulation process, each element of the input vector is read from an external file by the CSC and broadcast over the Inbus to all four PNs. PN0 performs the multiply $v_0 * w1_{00}$; PN1, $v_0 * w1_{10}$; etc. This happens in one clock. In the next clock, v_1 is broadcast, PN0 computes $v_1 * w1_{01}$, PN1, $v_1 * w1_{11}$, etc. Meanwhile, the previous clock's products are sent to the adder, which contains zero initially.

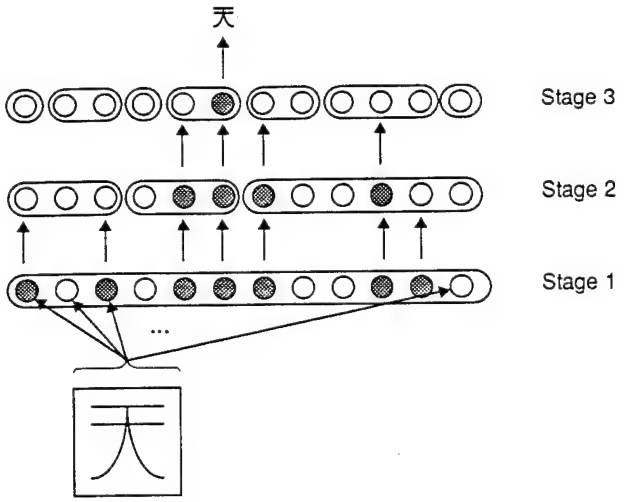
All hidden-layer products have been generated after five clocks. One more clock is required to add the last product to the accumulating sum (ignoring the bias terms here for simplicity). Next, all PNs take the most-significant byte out of the product and use it as an address into the lookup table to get the sigmoid output. The read value then is put into the output buffer, and the PNs are ready to compute the output node outputs.

The next step is computing the output-layer node values (nodes 4 and 5). In the first clock, PN0 transmits its output (node 0's output) onto the output bus. This value goes through the CSC and comes out on the Input bus, where it is broadcast to all PNs. Although only PN0 and PN1 are used, all PNs compute values (PN2 and PN3 compute dummy values). PN0 and PN1 compute $n_0 * w2_{00}$ and $n_0 * w2v_{01}$. In the next clock, node 1's value is broadcast and $n_1 * w2_{10}$ and $n_1 * w2_{11}$ are computed, and so on. After four clocks, PN0 and PN1 have computed all products. One more clock is needed for the last addition; then, a sigmoid table lookup is performed. Finally, the node 4 and 5 outputs are transmitted sequentially on the Outbus, and the CSC writes them into a file.

Let a *connection clock* be the time it takes to compute one connection. For standard BP, a connection requires a multiply-accumulate plus, depending on the architecture, a memory fetch of the next weight, the computation of that weight's address, etc. For the CNAPS PN, a connection clock takes one cycle. On a commercial microprocessor chip, a connection clock can require one or more cycles, since many commercial chips cannot simultaneously execute all operations required to compute a connection clock: weight fetch, weight address

Figure 12.

A schematicized version of the three layer LVQ network that Sharp uses in their Kanji OCR system. The character is presented as a 16x16 or 256 element system. Some characters are recognized immediately; others are merely grouped with similar characters, ©IEEE.



increment, input element fetch, multiply, and accumulate. These operations can take up to 10 clocks on many microprocessors. Much of this overhead is memory fetch, since many state-of-the-art microprocessors are making more use of several levels of intermediate data caching. And, as discussed previously, ANNs are notorious cache busters, so many memory and input element fetches can take several clocks each.

Simulating a three-layer BP network with N_I inputs, N_H nodes in the hidden layer, and N_O nodes in the output layer will require $(N_I * N_H) * (N_H * N_O) + N_O$ connection clocks for non-learning, feed-forward operation on a single processor system. On CNAPS, assuming there are more PNs than hidden or output nodes, the same network will require $N_I + N_H + N_O$ connection clocks. For example, assume that $N_I = 256$, $N_H = 128$, and $N_O = 64$. For a single processor system, the total is 73,792 connection clocks; for CNAPS, 448. If a workstation takes about four cycles on average, which is typical, to compute a connection, then CNAPS is about 600x faster on this network.

Back-Propagation: Learning Phase

The second and more complex aspect of BP learning is computing the weight delta for each connection. A detailed discussion of this computation and its CNAPS implementation is beyond the scope of this paper, so only a brief overview is given here. The computation is more-or-less the same as a sequential implementation. The basic learning operation in BP is to compute an error signal for each node. The error signal is proportional to that node's contribution to the output error (the difference between the target output vector and the actual output error). From the error signal, a node can then compute how to update its weights. At the output layer, the error signal is the difference between the feed-forward output vector and the target output vector for that training vector. The output nodes can compute their error signals in parallel.

The next step is to compute the delta for each output node's input weight (the hidden-to-output weights). This computation can be done in parallel, with each node computing, sequentially, the deltas for all weights of the output node on this PN. If a batching algorithm is used, then the deltas are added to a data element associated with each weight. After several weight updates have been computed, the weights are updated according to an accumulated delta.

The next step is to compute the error signals for the hidden-layer nodes, which requires a multiply-accumulate of the output-node error signals through the output-node weights. Unfortunately, the output-layer weights are in the wrong place (on the output PNs) for computing the hidden-layer errors. That is, the hidden nodes need weights that are scattered among the output PNs, which can best be represented as a transpose of the weight matrix for that layer. A transpose operation is slow on CNAPS, taking $O(N^3)$ operations. The easiest solution was to maintain two weight matrices for each layer, the feed-forward version and a transposed version for the error back-propagation. This requires twice the weight memory for each hidden node, but permits error propagation to be parallel, not serial. Although the new weight value need only be computed once, it must be written to two places. This duplicate weight matrix is required only if learning is to be performed.

After the hidden-layer error signals have been computed, the weight delta computation can proceed exactly as described above. If more than one hidden layer is used, then the entire process is repeated for the second hidden layer. The input layer does not require the error signal.

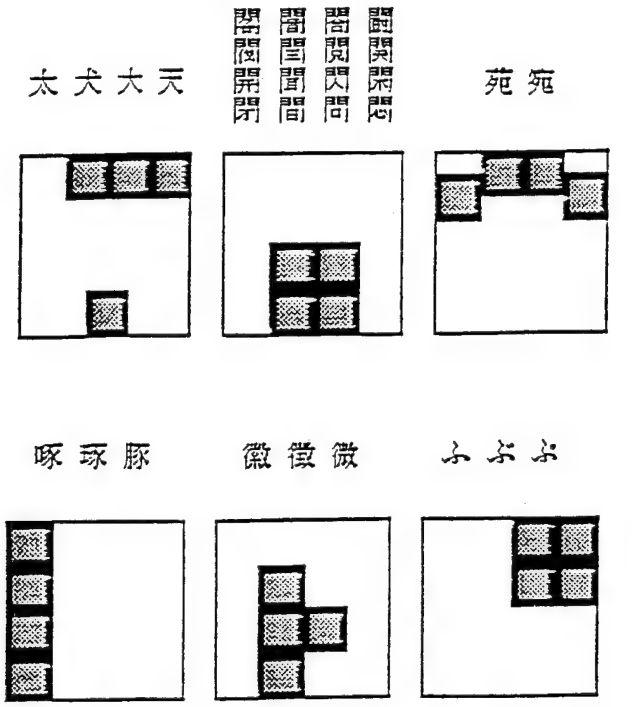
For non-batched weight update, where the weights are updated after the presentation of each vector, the learning overhead requires about five times more cycles than feed-forward execution. A 256 PN (four chip) system with all PNs busy can update about one billion connections per second, almost one thousand times faster than a Sparc2 workstation. A BP network that takes an hour on a Sparc2 takes only a few seconds on CNAPS.

Simple Image Processing

One major goal of CNAPS was flexibility because, by Amdahl's law, the more of the problem that can be parallelized, the better. Therefore, other, parallelizable, but non-ANN, parts of the problem should also be moved to CNAPS where possi-

Figure 13.

Distinguishing members of a group by focusing on a group specific subfield. Here a more detailed 32 x 32 image is used, ©IEEE.



ble. Many imaging applications, including OCR programs, require image processing before turning the ANN classifier loose on the data. A common image processing operation is convolution by spatial filtering.

Using spatial (pixel) filters to enhance an image requires more complex computations than simple pixel operations require. Convolution, for example, is a common operation performed during feature extraction to filter spatial noise or define edges. Here, a kernel, an M by M dimensional matrix, is convolved over an image. In the equation below, for instance, the local kernel, k, is convolved over an N by N image, a, to produce a filtered N by N image b:

$$b_{i,j} = \sum_{p,q} k_{p,q} a_{i-p,j-q}$$

$$(i \le i,j \le N)\ (1 \le p,q \le M)$$

Typical convolution kernels are Gaussian, differences of Gaussian, and Laplacian filters. Due to their inherent parallelism, convolution algorithms can be easily mapped to the CNAPS architecture. The image to be filtered is divided into regions called "tiles". One or more tiles are mapped to each PN. The kernel values, k, are then broadcast to the array. If the image does not fit in the PN array, then only a subset of the tiles are moved in and computed at a time.

There are two ways to deal with the fact that the convolutions will overlap at the edges of tiles: send in overlapping tiles, or send values in tile edges to neighboring PNs (that have neighboring tiles) via the inter-PN bus before each convolution operation. The first method is used if only one convolution is performed on a tile before outputting it again. This method is used by our PhotoShop accelerator product, which will be discussed below. The second method is used by general image processing routines that do several operations on an image before it is output.

For neural networks, the problem data is the same for each node in the network and the coefficients (weights) are different. This led to allocating the coefficients the PNs and broadcasting the data. However, because of the sparseness of the convolution, it is more efficient to put the data (the image tiles) on the PNs and then broadcast the coefficients.

Because of the parallel structure of this algorithm, all PNs can calculate the convolution kernel at the same time, convolving all pixels in one row simultaneously. Using different kernels, this convolution process can be carried out several times, each time with a different type of spatial filtering performed on the image.

For a 18MB image in full color, RGB 3-byte per pixel representation, a 7x7 convolution kernel can be performed by a single 64-PN chip in about 7 seconds. This includes the time to move the image onto the CNAPS board and off again.

Naval Air Warfare Center

At the Naval Air Warfare Center (NAWC) at China Lake, California, ANN technology has been aimed at air-launched tactical missiles. Processing sensor information on board these missiles demands a computational density (operations per second per cubic inch) far above most applications. Tactical missiles typically have several high-data-rate sensors, each with its own separate requirements for high-speed processing. The separate data must then be fused, and the physical operation of the missile controlled. All this must be done under millisecond or microsecond time constraints and in a volume of a few cubic inches. Available power is measured in tens of watts. Such immense demands have driven NAWC researchers toward ANN technology.

For some time (1986 to 1991), many believed that analog hardware was the only way to achieve the required computational density. The emergence of wafer scale, parallel digital processing (exemplified by the CNAPS chip) has changed that assessment, however. With this chip, we have crossed the threshold at which digital hardware – with all its attendant

flexibility advantages – has the computational density needed to be useful in the tactical missile environment. Analog VLSI may still be the only way to overcome some of the most acute time-critical processing problems on board the missile, for example, at the front end of an image processing system. A hybrid system combining the best of both types of chips may easily turn out to be the best solution.

Researchers at NAWC have worked with several versions of the CNAPS system. They have easily implemented corticomorphic computational structures on this system – structures that were difficult or impossible under the analog constraints of previous systems. They have also worked with Adaptive Solutions to design and implement a multiple-controller CNAPS system (a multiple SIMD architecture or MSIMD) with high-speed data-transfer paths between the subsystems. And they are completing the design and fabrication of a real-time system interfaced to actual missile hardware. The current iteration will be of the SIMD form, but the follow-on will have the new MSIMD structure.

The prototype system, called MAVIS (Missile borne Artificial VIsion System), is accurate and effective, but is also compute intensive. As input, the system takes images of 128x128 pixels, with 8 bits per pixel, at 60 images per second. The algorithm discussed above requires about 12,000 operations per pixel per frame or 11 billion operations per second. This system must fit inside a missile and consume only a moderate amount of power. This implementation uses three 256 PN CNAPS VME cards and operates in real time. With the current CNAPS implementation technology, it would be possible to put the system in the target platform by using a Multi-Chip Module with a ceramic substrate.

One important near-term application of this computational structure is in the area of adaptive, non-uniformity compensation for staring focal plane arrays. It appears also that this structure will allow the implementation of three dimensional wavelet transforms where the third dimension is time.

Lynch/Granger Pyriform Implementation

Researchers Gary Lynch and Richard Granger at the University of California, Irvine have produced an ANN model based on their studies of the Pyriform cortex of the rat. The algorithm contains features abstracted from actual biological operation, and has been implemented on the CNAPS parallel computer [13].

The algorithm contains both parallel and serial elements, and lends itself well to execution on CNAPS. Clusters of competing neurons, called "patches" or "subnets," hierarchically classify inputs by first competing for the greatest activation within each patch, then subtracting the most prominent features from the input as it proceeds down the lateral olfactory tract (the LOT, the primary input channel) to subsequent patches. Patch activation and competition occur in parallel in the CNAPS implementation. A renormalization function analogous to the automatic gain control performed in Pyriform cortex also occurs in parallel across competing PNs in the CNAPS array.

Transmission of LOT input from patch-to-patch is an inherently serial element of the Pyriform model, so opportunities for parallel execution for this part of the model are few. Nevertheless, overall speedups for execution on CNAPS (compared to execution on a serial machine) of 50 to 200 times are possible, depending on network dimensions.

Refinements of the Pyriform model and applications of it to diverse pattern recognition applications continue.

Sharp Kanji

Another application that has successfully used ANNs and the CNAPS system is a Kanji optical character recognition (OCR) system developed by the Sharp Corporation of Japan. In OCR, a page of printed text is scanned to produce a bit pattern of the entire image. The OCR program's task is to convert the bit pattern of each character into a computer representation of the character. In the US and Europe, the most common representation of Latin characters is the 8-bit ASCII code. In Japan, because of their unique writing system, it is the 16-bit JIS code.

OCR requires a complex set of image recognition operations. Many companies have found that ANNs are effective for OCR because ANNs are powerful classifiers. Many commercial OCR companies, such as Caere, Calera, Expervision, and Mimetics, use ANN classifiers as a part of their application software.

Japanese OCR is much more difficult than English OCR because Japanese has a larger character set. Written Japanese has two basic alphabets. The first is Kanji, or pictorial characters borrowed from China. Japanese has tens of thousands of Kanji characters, although it is possible to manage reasonably well with about 3500 characters. Sharp chose these basic Kanji characters for their recognizer.

The second alphabet is Kana, comprised of two phonetic alphabets (Hiragana and Katakana) having 53 characters each. Typical written Japanese mixes Kanji and Kana. Written Japanese also employs arabic numerals and Latin characters, typically found in business and newspaper writing. A commercial OCR system must be able to identify all four types of characters. To add further complexity, any character can appear in several different fonts.

Japanese keyboards are difficult to use, so a much smaller proportion of business documentation than one sees in the United States and other western countries is in a computer readable form. This difficulty creates a great demand for the ability to accurately read printed Japanese text and convert it to the corresponding JIS code automatically. Unfortunately, due to the large alphabet, the computer recognition of written Japanese is a daunting task. At the time this paper is being written, the commercial market consists of slow (10-50 char-

acters per second), expensive (tens of thousands of dollars), and marginally accurate (96%) systems. Providing high speed and accuracy for a reasonable price would be a quantum leap in capability in the current market.

Sharp Corporation and Mitsubishi Electric Corporation have both built prototype Japanese recognition systems based on the CNAPS architecture. Both systems recognize a total of about 4000 characters in 50 different fonts at accuracies of over 99% and speeds of several hundred characters per second. These applications have not yet been released as commercial products.

Sharp's system uses a hierarchical three-layer network, [8] and [20], (Figures 12 and 13). Each layer is based on Kohonen's Learning Vector Quantization (LVQ) algorithm, a Bayesian approximation that shifts the node boundaries to maximize the number of correct classifications. In Sharp's system, unlike back-propagation, each hidden-layer node represents a character class, and some classes are assigned to several nodes. Ambiguous characters pass to the next layer. When any layer unambiguously classifies a character, it has been identified, and the system moves on to the next character.

The first two levels take as input a 16x16 pixel image (256 elements). With some exceptions, these layers classify the character into multiple subcategories. The third level has a separate network per subcategory. It uses a high-resolution 32x32 pixel image (1024 elements), focusing on the subareas of the image known to have the greatest differences among characters belonging to the subcategory. These subareas of the image are trained to tolerate reasonable spatial shifting without sacrificing accuracy. Such shift tolerance is essential due to differences among fonts and shifting during scanning.

Sharp's engineers clustered 3303 characters into 893 subcategories containing similar characters. The use of subcategories let Sharp build and train several small networks instead of one large network. Each small network took its input from several local receptive fields designed to look for particular features. The locations of these fields were chosen automatically during training to maximize discriminative information. The target features are applied to several positions within each receptive field, enhancing the shift tolerance of the field.

On a data base of scanned characters that included more than 26 fonts, Sharp reported an accuracy of 99.92% on the 13 fonts used for training and 99.01 percent accuracy on characters on the 13 fonts used for testing. These results show the generalization capabilities of this network.

Photoshop Acceleration

The "Prepress" market segment involves activities that occur between the development of an electronic document and its preparation for printing. The most complex preparation tasks concern photographs, figures, and other kinds of complex images in the document. The most popular program for manipulating photographs is Adobe Photoshop.

Most commercial images (in magazines, advertising brochures, and similar publications) contain are large, typically tens of megabytes. Simple image processing functions, such as a 2D spatial filter, can be slow, even on high speed computers. Because the performance-cost of CNAPS exceeds that of traditional desktop computers, Adaptive Solutions decided to build a Photoshop CNAPS accelerator card, "PowerShop." The small, simple CNAPS PN arrays and the use of redundancy allows us to offer such performance at an affordable price. In addition, the these images typically use low precision, integer data representations which also maps efficiently to CNAPS.

Photoshop uses several filters. These filters are implemented like the convolution discussed above. Since Photoshop updates the displayed version of the image in the host memory after each operation, the CNAPS card reads an image, a single operation is performed, then the image is written back to main memory. For this reason a two-level tiled version of the convolution algorithm is used. The image is broken up into large tiles, each the size of a PN array. CNAPS then processes one tile at a time, sequentially reading a tile, then writing an updated version of the tile back to main memory.

In general, the performance improvements of the CNAPS PowerShop card over a Power Mac (PCI bus based) range from factors of 3-10x. A 7x7 convolution filter over a 24-bit full color 18 MB image using a 64 PN array is about 7 seconds, versus 89 seconds on a PowerMac 8100.

Medical Image Processing

An important area of image processing and pattern recognition concerns the classification medical images, a field that has significant computing requirements and increasing pressure to decrease costs. Reading and analyzing scanned images–whether MRI scans, optical scans such as Pap smears, or X-rays for suspicious structures such cancer cells–is a matter of life or death. The data is noisy and ambiguous, and is error prone. R2 Technology has developed a neural network based classification algorithm for identification areas of interest in mammograms.

The R2 application uses a combination of standard image processing techniques for image preprocessing and then a neural network algorithm for the final classification. The CNAPS PCI board with 128 PNs can scan an entire 4K x 4K X-ray in 14 seconds, which meets R2's performance requirements.

Conclusion

This paper has given only a brief view into a commercial ANN product and into the decisions made during its design. It has also briefly examined some real applications that use this product. The reader should have a better idea about why the various design decisions were made during this process and the final outcome of this effort. The CNAPS system has

achieved its goals in speed and performance and, as discussed, is finding its way into real world applications.

Acknowledgements

I would like to acknowledge the following people and their contributions to the paper: Dave Andes of the Naval Air Warfare Center, and Eric Means and Steve Pendleton of Adaptive Solutions.

The Office of Naval Research funded the development of the implementation of Lynch/Granger model on the CNAPS system.

Biography

Dr. Hammerstrom is on the faculty of the Computer Science and Engineering Department at the Oregon Graduate Institute and is the founder and Chief Technical Officer of Adaptive Solutions Inc. After getting a Ph.D. in electrical engineering from University of Illinois, he was on the faculty of Cornell from 1977 to 1980. He then worked for Intel Corporation where he deigned the architecture of the following milestone microprocessors: iAPX-432, the i960, and the iWarp. He is an Associate Editor for the IEEE Transactions on Neural Networks and the International Neural Networks.

References

[1] L.A. Akers, S. Haghighi, and A. Rao. Vlsi implementations of sensory processing systems. In *Proceedings of the Neural Networks for Sensory and Motor Systems (NSMS) Workshop,* March 1990.

[2] Joshua Alspector and el al. Experimental evaluation of learning in a neural microsystem. In *Advances in Neural Information Processing Systems III.* Morgan Kaufman, 1991.

[3] Tom Baker and Dan Hammerstom. Characterization of artificial neural network algorithms. In *1989 International IEEE Sumposium on Circuits and Systems*, pages 78-81, September 1989.

[4] M. Griffin et al. An 11 million transistor digital neural network execution engine. In *The IEEE International Solid State Circuits Conference,* 1990.

[5] Hans P. Graf, Lawrence D. Jackel, and Wayne E. Hubbard. Vlsi implementation of a neural network model. *IEEE Computer*, 21(3):41-49, 1988.

[6] D. Hammerstrom. A vlsi architecture for high-performance, low-cost, on-chip learning. In *Proceedings of the IJCNN, 1990.*

[7] D. Hammerstrom. A highly parallel digital architecture for neural network emulation. In J.G. Delgado-Frias and W.R. Moore, editors, *VLSI for Artificial Intelligence & Neural Networks.* Plenum Press, 1991.

[8] Dan Hammerstrom. Neural networks at work. In *IEEE Spectrum*, pages 26-322, June 1993.

[9] M. Hoehfeld and S.E. Fahlman. Learning with limited numerical precision using the cascade-correlation algorithm. *IEEE Transactions on Neural Networks,* 3(), July 1992.

[10] Mark Holler, Simon Tamm, Hernan Castro, and Ronald Benson. An electrically trainable artificial neural network (etann) with 10,240 floating gate synapses. In *Proceedings of the IJCNN,*) 1989.

[11] H. McCartor. Back-propagation implementations on the adaptive solutions neurocomputer chip. In *Advances in Neural Information Processing Systems II,* Denver, CO, 1991. Morgan Kaufman.

[12] Carver Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, New York NY, 1989.

[13] Eric Means and Dan Hammerstrom. Piriform Model execution on a neurocomputer. In *Proceedings of the IJCNN*, 1991.

[14] R.W. Means and L. Lisenbee. Extensible linear floating point simd neurocomputer array processor. In *Proceedings of the IJCNN*, 1991.

[15] Nelson Morgan, editor. *Artificial Neural Networks: Electronic Implementations.* Computer Society Press Technology Series and Computer Society Press of the IEEE, Washington, D.C., 1990.

[16] U. Ramacher and et al. Multiprocessor and memory architecture of the neurocomputer synapse-1. In *Proceedings of the World Congress of Neural Networks,* 1993.

[17] D. Rumelhart and J. McClellan. *Parallel Distributed Processing.* MIT Press, New York, NY, 1986.

[18] T. Sejnowski and C. Rosenberg. Nettalk: A parallel network that learns to read aloud. Technical Report JHU/EECS-86/01, The Johns Hopkins University Electrical Engineering and Computer Science Department, 1986.

[19] P.A. Shoemaker, M.J. Carlin, and R.L. Shimabukuro. Back-propagation learning with coarse quantization of weight updates. In *Proceedings of the IJCNN*, 1990.

[20] F. Togawa, T. Ueda, T. Aramaki, and A. Tanaka. Receptive field neural networks with shift tolerant capability for Kanji character recognition. In *Proceedings of the International Joint Conference on Neural Networks,* June 1991.

[21] J. Wawrzyneck, K. Asanovic, and N. Morgan. The design of a neuro-microprocessor. *IEEE Transactions on Neural Networks*, 4(3), May 1993.

Research Notes

An Intelligent Chip

The fastest neural network processor, the Ni1000 computer chip, was developed recently by Intel Corp. of Santa Clara, California. The new technology is the most promising approach toward building intelligent machines that mimic hearing, seeing and thinking. The chip is amazingly quick at recognizing handwriting, identifying military targets and performing other tasks that are difficult or impossible for conventional chips. There are numerous civilian applications for this chip, including finger print identifications, automatic mailing address processing, and even stock market forecasting and predictions.

The new Ni1000 chip, developed with funding from the Advanced Research Projects Agency (ARPA) and the Office of Naval Research (ONR), is an unusual breed called neural networks. These chips work more like the human brain than the microprocessors used in millions of personal computers. Because they can recognize visual or sound patterns at high speed, neural nets are being applied to tricky tasks such as distinguishing human voices and zip codes. ARPA is interested in these chips for identifying submarines and other targets.

Intel's new chip is expected to be particularly useful in handwriting recognition, a rapidly growing market, and will be a hundred times faster than other technologies used for that purpose. Nestor Inc. of Rhode Island developed a version of the handwriting-recognition algorithm for the Ni1000. A scanner based on a fast version of Intel's 486 microchip can recognize about 30 handwritten characters per second while the Ni1000 is expected to recognize 5,000 to 10,000 characters. Although the chip requires too much electric current for use in small computers, Intel is working to improve the chip for use in hand-held machines.

Where other chips answer precise mathematical questions, neural net chips can be trained to work on more subjective problems. Interconnected processing elements on each chip, called neurons, join in different ways when exposed to different signals. By employing a large number of processing elements that operate in parallel, the Ni1000 performs 20 billion interconnection operations per second. The chip uses a large block of flash memory so that learned patterns can be "memorized" and quickly "recalled" for real-time pattern recognition applications. Learning capability is implemented on-chip in the form of a 16-bit microcontroller.

"The Ni1000 chip represents a new generation of highly intelligent, high performance chips based on the neural network computations paradigm," said Dr. Clifford Lau, acting director of the Electronics Division at ONR and the scientific officer overseeing ONR's participation in the chip development

ONR has a long history of supporting neural networks research. In the 1950's ONR funded the research of F. Rosenblatt on the perceptron, which is now the basic processing element of multilayer perceptron neural networks. In the 1960's, ONR supported the research of Professor B. Widrow at Stanford University on the adaptive linear neuron, or ADALINE, together with the least mean square adaptation algorithm, which now forms the basis for the popular back propagation learning algorithm in artificial neural networks. ONR recognized in the 1980's the importance of understanding how the brain processes and stores information, and started to invest in research on learning and memory. The objectives of ONR's programs today according to Dr Lau are "to understand the architectures of the brain and the algorithms for brain information processing, and to formulate computational neuroscience models.

Dr. Leon Cooper, a long time ONR principal investigator and Nobel laureate said "Combining neural networks that learn and capture the human ability for rapid pattern recognition with the processing power of personal computers will bring us to the next generation of decision-making machines.

A Bionic Eye

"A computer-packed bionic eye may soon match the sensitivity of human and animal eyes," say Professor Leon O. Chua of the University of California at Berkeley and Professor Tomas Roska of the Hungarian Academy of Sciences at Budapest. Their research program, which is another case of science trying to imitate nature, is sponsored by the National Science Foundation and the Office of Naval Research. At ONR the Scientific Officers overseeing the work are Dr. Clifford Lau of the Systems and Electromagnetic Theory Division and Dr. Joel Davis of the Computational Neuroscience Division.

This bionic eye is part of the popular trend of combining the disciplines of biologists and computer scientists to endow machines with intelligence and senses. The scientists are working toward a supercomputer etched into a thumbnail-size chip on which an image is focused. The computer-eye will be made to see like a cat, salamander, hawk, or person. The nerve circuitry will be able to pick out some things and pay less attention to others.

Eyes include layers of densely packed neurons, literal extensions of the brain, that sort the image on the retina into lines, corners, shades of color and gray, edges and moving objects even before the image goes to the brain for more

abstract analysis. "The thing is, eyes don't work like a camera, just focusing and recording images," said Professor Frank Werblin of Berkeley, who has recently joined the bionic eye team. Werblin is known for his pioneering work on the densely packed nerves in the retinas of salamanders and uses a conventional computer the size of a refrigerator to mimic the vision in a salamander's eye. The bionic eye project hopes to put the entire computer on a chip of silicon smaller than a thumbnail but with the potential calculating speed of the biggest super computers: a trillion calculations per second.

Roots to the bionic eye go back to a tiny, computer-on-a-chip called the cellular neural network (CCN), that Chua developed and is now refining with funding from ONR under the supervision of Dr. Rabinder Madan of the System and Electromagnetic Theory Division. Chua has applied for a patent, naming the device the "CNN Universal Machine." The CNN contains hundreds to thousands of interconnected, identical "cells" each one a simple bit of circuitry connected to its nine closest neighbors. Properly set up so that each cell knows directly only what it and its neighbors see, but with all the cells cooperating and calculating at once, information surges back and forth at lightening speed. If programmed to see only straight lines shorter than a certain length, such a device would immediately spot flaws in woven fabric.

Some of the layers in salamander retinas see only moving objects, other see objects only of certain sizes, and other look for edges. In frogs, the system blinds the animals to almost everything except objects of the same size and motion as the flies and moths they snare with their tongues For salamanders and most vertebrates, eyes work the same way. The presorted images get to the brain's visual cortex where another dozen or so additional layers of brain tissue further break images down. A single neural network, however, could take the place of all the layers of neurons, switching rapidly from one mode to another.

Way down the road, perhaps many decades from now, there could be truly bionic eyes as on TV shows, where badly injured people are turned into part-machine superheroes. Artificial eyes could let the blind see by hooking directly into optic nerves or even brains, but nobody knows how to do that yet.

Soon, scientists boldly predict that the computer-eye might recognize wanted criminals or lost children, instantly detect flaws in manufactured goods, identify targets for automated military weapons or recognize mineral deposits from space.

The bionic eye team is learning the truth of the old saying, "Beauty is in the eye of the beholder."

A Computer with an IQ

When Professors Richard Granger and Gary Lynch at the University of California, Irvine, duplicated six years ago brain circuitry in a computer program, they had no idea that it would begin acting like a brain. They mapped the circuits of a small piece of rat brain and then duplicated the circuits in a computer program, "just to see what would happen." Last year, the Office of Naval Research (ONR) tested with great success the program for relevant Navy use, such as recognizing sonar signals.

Soon after creating the program, Granger and Lynch started feeding their computer signals, simulations of the electrical impulses chemical stimulants create in the brain. Not too surprisingly, the computer stored memory of the stimuli as the brain does and could recognize them when it perceived them again. One night while Lynch was playing the program, the computer did a new trick. When it was fed a simulated "odor," it not only sent back the recognition signal, it sent back a preliminary signal as well. Granger and Lynch were amazed when they realized months later that the second signal denoted a category, a grouping of similar odors that the computer had devised all on its own. Without being told to do it, the computer had grouped all flower smells together and all cheese smells together. "It had spontaneously reproduced a psychological process." Lynch says, "because that's how the brain circuits are designed to operate. You and a rat and every mammal do it without thinking.

When the computer memorized enough chemical stimuli and put them into categories, it performed some sophisticated recognition. It detected odors masked by stronger and different chemical stimuli. It would say, "that's roses, and there's a magnolia and some cheddar in there, too." Once the computer was wired like a brain, it acted like a brain. You could not instruct it to record or sort odors. You could only present it with stimuli and let it do what it pleased with them.

A human brain has 10 billion brain cells or neurons, but the computer has only 1,000 simulated neurons. Lynch claims that the computer learned to recognize 10,000 words. " If we could build a model with 100,000 neurons, we could have taught it a new word every five seconds for 50 years; it would be eager for more and categorizing them."

When ONR tested this "thinking" computer, it mastered difficult classifications which involved real ocean passive acoustic signals; the computer recognized 95 percent of the signals and gave no false alarms. The best records before had been 25 percent and 60 percent on two other systems.

Dr. Joel Davis, the ONR scientific officer who has funded this work since its inception, says, "The Lynch-Granger program follows biological patterns more closely than any previous neural program. It's strongest where traditional programs are weakest – recognizing complex patterns. Besides classifying sonar signals, it might recognize the vibration patterns of mechanical parts about to fail and give warning. This field is in its infancy."

Perhaps HAL, the think feeling computer in the movie "2001" is a possibility for the not too distant future.